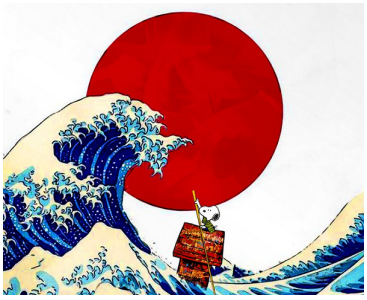
TP 9
Images en couleur – Sujet

Avant toute chose, créer un répertoire TP09 dans votre répertoire "Informatique", "TPsup" dans votre espace personnel. Sur le site `https://ptsilamartin.github.io/info/TP.html`, télécharger les 3 images `vague.png`, `groupe.png` et `IRM.png` ainsi que l'animation gif `battements` que vous copiez dans votre répertoire TP09.



Dans ce TP, comme dans le TP08 on utilise images matricielles. Contrairement au TP08 les images sont des images colorées. La teinte de chaque pixel est codée sous la forme d'un tableau `[R,G,B]`, `R` codant la valeur du rouge, `G` du vert et `B` du bleu. `R`, `G`, `B` sont des nombres réels compris entre `0` et `1`.

Accentuation d'une composante de couleur

Question 1 Lire l'image du fichier `vague.png` et l'afficher.

Question 2 Ecrire une fonction `accentuation` qui prend pour argument une matrice-image codée au format RGB et un flottant `k`, et qui accentue de `k` la valeur de la composante rouge de chaque pixel. Afficher l'image obtenue avec une valeur de `k` de 0.2.

Conversion d'une image couleur en niveau de gris

Une des possibilités pour obtenir une image en niveau de gris à partir d'une image codée sur les 3 composantes de couleur (bleu, vert, rouge) est de remplacer la valeur de chaque composante de chaque pixel par la moyenne des valeurs des 3 composantes du pixel considéré. Cette méthode est simple mais peu convaincante au niveau de l'image.

La CIE (Commission Internationale de l'Eclairage) normalise la répartition pour obtenir un niveau de gris correct. Dans sa norme 709, il est indiqué que pour les images naturelles les poids respectifs doivent être : $0.2125 * R + 0.7154 * G + 0.0721 * B$.

Rappel : pour afficher une image en niveau de gris utiliser la fonction : `plt.imshow(im, cmap="gray", vmin=0, vmax=1)`

Question 3 Ecrire une fonction `niveau_gris` qui prend pour argument une matrice-image codée au format RGB et qui convertit cette image en niveau de gris en utilisant la moyenne des composantes.

Question 4 [OPTIONNELLE : uniquement si vous êtes en avance] Ecrire une fonction `niveau_gris2` qui convertit une image en niveau de gris en utilisant la norme 709.

Flou du visage d'une personne sur une image

Pour des questions de protection de la vie privée, il est courant de devoir flouter les visages des personnes qui ne souhaitent pas apparaitre sur une photo.

Flou avec une matrice 3x3

Question 5 Lire et afficher l'image `groupe.png`.

Question 6 Créer une fonction `flou3` qui prend pour argument une image `im` et qui retourne l'image floutée en utilisant la matrice 3x3 de filtrage définie dans le cours.

Pour cela :

- déterminer : la hauteur et la largeur de l'image ainsi que le nombre de composantes par pixel ;
- créer une image vide ne contenant que des zéros ;
- pour chaque pixel (hors pixels situés sur les bords de l'image), et pour chaque composante du pixel considéré : affecter la valeur calculée à partir de la matrice de filtrage et des voisins.

Question 7 Tester sur l'image proposée. On obtient un effet de flou très (trop) léger.

Flou avec une matrice nxn

Pour accentuer l'effet de flou, on propose d'utiliser une matrice de filtrage **n x n** ne contenant toujours que des 1.
Il faudra donc exclure du traitement une bande de pixels de largeur $b = n//2$ sur chaque bords de l'image.

Question 8 Créer une fonction `flou` qui prend pour argument une image `im` ainsi que la taille de l'image matrice filtre `n` et qui renvoie l'image floutée.

Question 9 Tester sur l'image proposée, pour `n=7` par exemple.

Flou sur une zone de l'image seulement

Dans cette partie on propose de ne flouter qu'une partie de l'image, autour d'un visage par exemple. Pour cela on utilise un masque définissant la zone à traiter.

Ce masque sera une image toute noire, sauf les pixels à traiter qui seront en blanc.

Question 10 Créer une matrice image masque de la taille de l'image à traiter ne contenant que des pixels noirs, sauf pour les pixels situés sur les lignes allant de 80 à 179, et sur les colonnes allant de 370 à 449. Ces pixels seront blancs.

Attention : l'image masque doit avoir les mêmes caractéristiques que l'image traitée : la valeur d'un pixel blanc sera `[1,1,1,1]` et celle d'un pixel noir `[0,0,0,1]`, si l'image est codée sur 4 composantes (c'est le cas pour format **.png** qui gère la transparence en plus des 3 ccomposantes RGB).

A partir du masque, de l'image floutée et de l'image initiale, on peut obtenir ainsi l'image finale :

$$\text{ImageFinale}_{i,j,c} = \text{Masque}_{i,j,c} \times \text{ImageFloue}_{i,j,c} + (1 - \text{Masque}_{i,j,c}) \times \text{ImageInitiale}_{i,j,c}$$

où i et j définissent la position du pixel et c la composante considérée.

Question 11 À partir de la matrice masque, de la matrice de l'image initiale, et la matrice de l'image floutée, générer une image où seul le visage défini par le masque est flouté. L'afficher.

Remarque

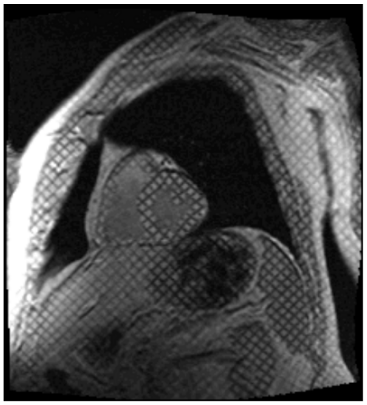
Les images sont des tableaux `Numpy` : on pourra donc utiliser les propriétés bien pratiques des tableaux `Numpy`.
Il est ainsi possible de réaliser directement des opérations sur ces tableaux Numpy : ces opéraions seront comprises comme des opérations "composantes par composantes".
Exemple : `a=np.array([5, 2, 4]) ; b=np.array([4, 3, 6])`
Alors : `c=a*b` donnera `array([20, 6, 24])`

Pour aller + loin : Application à une problématique d'imagerie médicale

Les applications des algorithmes de traitement d'images sont multiples. On étudie ici l'utilisation de ces techniques pour le diagnostic et le suivi de pathologies cardio-vasculaires.
Source : Détection robuste et automatique des contours myocardiques sur des séquences IRM cardiaques marquées. Aymeric Histace, Christine Cavaro-Ménard. Les images ci-dessous proviennent toutes de cet article de recherche.

Objectif

L'objectif est ici d'automatiser la détection des contours du ventricule gauche afin de pouvoir étudier la contraction pariétale.



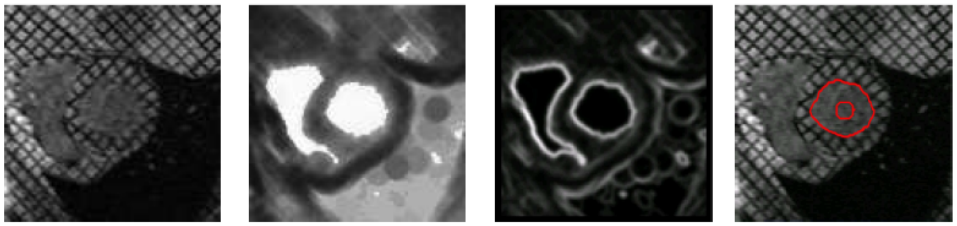
L'IRM cardiaque marquée permet de faire apparraitre de manière non invasive une grille (*tags*) sur la zone ventriculaire gauche. Ces *tags* disparaissent dans la cavité cardiaque : la cavité cardiaque est donc caractérisée par une zone homogène, la couronne mycardiqque, elle, est caractérisée par une texture en grille non homogène.

On voit ci-dessous l'image issue de l'IRM marqué (a).
L'image (b) est obtenue par création d'une carte de texture, dont chaque intensité de pixel s'obtient comme la somme pondérée de la moyenne (notée MOY) et de l'écart type (noté STD), d'un voisinage (de dimension T x T) autour du pixel considéré :

$$I(i,j) = 10.MOY(i,j) - 30.STD(i,j) \tag{0.1}$$

Cette approche permet d'augmenter de manière significative le niveau de gris des pixels situés dans les zones homogènes dont l'écart type, sur le voisinage considéré, est plus faible que pour les pixels proches des *tags*. Néanmoins, cette formule peut entrainer des valeurs négatives pour certains pixels : il conviendra alors de leur affecter une valeur nulle (noir).

Par un calcul de gradient on obtient l'image (c).
L'image (d) montre le contour endocardique recherché.



(a) (b) (c) (d)

Question 12 Ecrire les fonctions `Moyenne` et `EcartType` qui calculent respectivement la moyenne et l'écart type des valeurs des pixels d'une image.

Question 13 Ecrire une fonction `CarteTexture`, qui réalise, à partir d'une image en niveau de gris, la carte de texture (b) comme expliqué ci-dessus. Tester sur l'image fournie `IRM.png` . Vous choisirez judicieusement le paramètre T le plus adapté en réalisant éventuellement plusieurs essais.

Question 14 Déterminer le contour endocardique. Vous pourrez recadrer l'image autour de la zone souhaitée.