TP 04

Activités pratiques

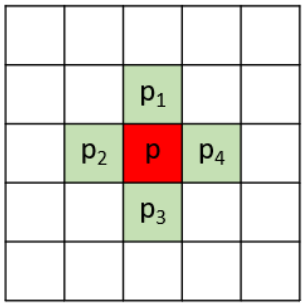
Avant toute chose, créer un répertoire TP04 dans votre répertoire "Informatique", "TPsup" dans votre espace personnel. Sur le site `https://mpsilamartin.github.io/info/TP.html`, télécharger les 3 images `vague.png`, `groupe.png` et `IRM.png` ainsi que l'animation gif `battements` que vous copiez dans votre répertoire TP04.

1 Détection de contours

La reconnaissance de formes dans une image est une composante importante de l'analyse d'images. Elle se décompose en plusieurs étapes qui consistent à extraire les contours des objets dans l'image afin de les reconnaitre ou d'en détecter le mouvement. La première de ces étapes est la mise en évidence des contours des objets dans l'image. C'est cette étape que nous allons aborder très succintement.

Un contour définit la limite d'un objet dans une image. Cette limite est caractérisée par un changement dans l'image : un changement de couleur ou de contraste. Ce changement se traduit dans la valeur des pixels qui sont localisés de part et d'autre de la limite. Nous sommes donc à la recherche d'un moyen de détecter et de localiser un changement.

Considérons un pixel **p** dans une image. Ce pixel est-il semblable, de même couleur, que ses voisins ? Si non, quelle est la différence de couleur entre lui et ses voisins ? Est-elle grande, ce qui signifierait qu'il est situé à la limite d'un objet ?
Le principe est pour chaque pixel de l'image, de récupérer la valeur de chaque pixel avoisinant ce pixel, puis de mesurer la différence, la "distance" euclidienne, entre ces valeurs en utilisant une fonction de norme standard :



$$\text{distance} = \sqrt{(p_1 - p_3)^2 + (p_2 - p_4)^2} \quad (1)$$

où p_1, p_2, p_3 et p_4 sont les valeurs des 4 pixels voisins.

Après avoir calculé la distance entre les valeurs des voisins du pixel courant, on compare cette distance à une valeur seuil :

- la distance est plus petite ou égale que le seuil : la variation est faible : ce pixel n'est pas considéré sur un contour : on le laisse en noir ;
- la distance est plus grande strictement que le seuil : la variation est grande : ce pixel est sur le contour : on le trace en blanc.

Pour pouvoir plus facilement comparer les "valeurs" des pixels, on travaillera sur une image en niveaux de gris. On rappelle qu'une image en niveaux de gris possède une dimension de moins qu'une image en couleur. En niveaux de gris, le gris est codé sur une valeur. En couleur, la couleur est codée sur 3 valeurs.

Question 1 *Lire l'image* `vague.png`. *Afficher cette image en utilisant le module* `matplotlib.pyplot`.

Question 2 *Comme réalisé dans l'activité préparatoire (vous pouvez réutiliser votre script), convertir cette image en niveaux de gris et l'afficher dans une nouvelle figure.*

Question 3 *Écrire une fonction* `contour`, *qui prend comme argument une image en niveaux de gris* `im` *ainsi qu'un flottant* `seuil`, *et qui renvoie une image noire sauf les pixels situés sur le contour qui seront en blanc.*

Question 4 *Tester cette fonction sur l'image précédemment obtenue, avec un seuil de 0.4.*

2 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.

2.1 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.*

2.2 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.*

2.3 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 :

$$\text{ImageFinale}_{i,j,c} = \text{Masque}_{i,j,c} * \text{ImageFloue}_{i,j,c} + (1 - \text{Masque}_{i,j,c}) * \text{ImageInitiale}_{i,j,c} \quad (2)$$

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

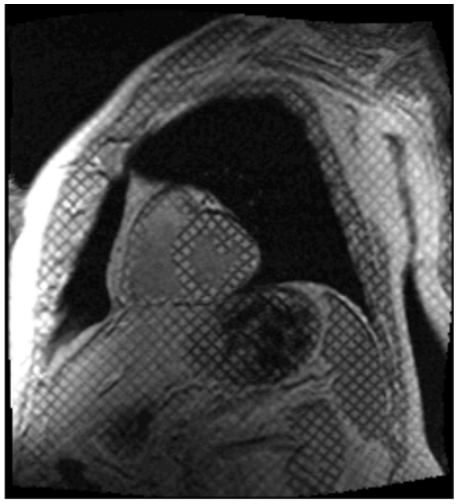
Question 11 *A partir de la matrice masque, de la matrice de l'image initiale, et la matrice de l'image floutée, générer une image où seul le visage défini par le masque est flouté. L'afficher.*

R 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])`

3 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.

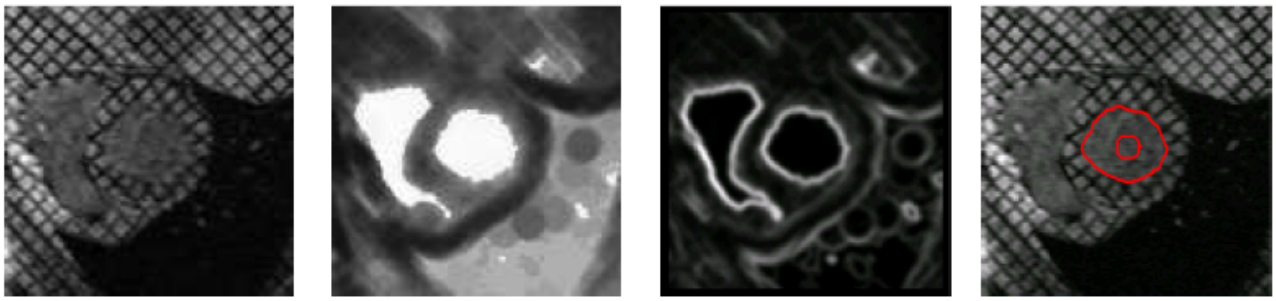
L'IRM cardiaque marquée permet de faire apparraitre de manière non invasive une grille (*tags*) sur la zone ventriculaire gauche. Ces *tags* disparaissent dans la cavité cardiaque : la cavité cardiaque est donc caractérisée par une zone homogène, la couronne mycardiqque, elle, est caractérisée par une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

$$I(i,j) = 10.MOY(i,j) - 30.STD(i,j) \tag{3}$$

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.



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.*