

TRAITEMENT D'IMAGERIE MEDICALES

MAÎTRISE EN INFORMATIQUE



IMN 503 : Devoir 1

JOUFFROY Emma (19157145) || ADOLPHE Maxime
(19156782)

Mr. DESCOTEAUX Maxime

Octobre 2019

Table des matières

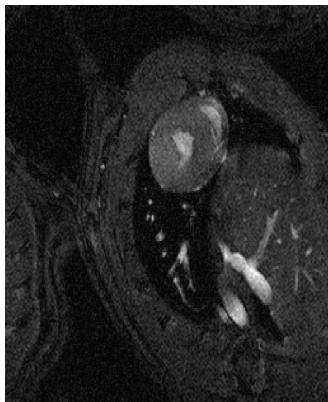
1 Question 1 : Développement d'un viewer	2
2 Question 2 : Exercice de manipulation des images	2
2.1 Taille des voxels et des images	2
2.2 Contraste de Michelson et RMS	3
2.3 Plus petite structure détectable et volume partiel	3
2.4 Bruit dans nos images	4
2.5 Calcul du SNR	6
3 Question 3 : Méthodes de débruitage	7
3.1 Ultrason	7
3.2 IRM Flair	7

1 Question 1 : Développement d'un viewer

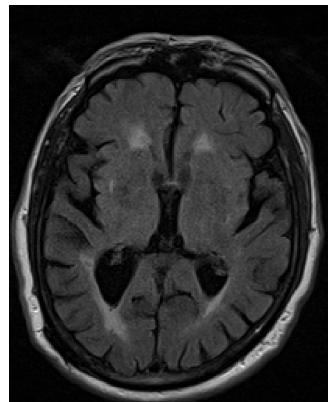
La première question de ce devoir a été réalisée dans le notebook fourni avec ce rapport. Vous trouverez des captures d'écran du travail effectué en annexe. Pour développer ce viewer, permettant d'afficher la coupe d'une image selon l'axe choisi, le langage Python a été utilisé, agrémenté de la bibliothèque de visualisation "holoviews".

2 Question 2 : Exercice de manipulation des images

La question 2 de ce devoir s'intéresse à la manipulation d'images médicales 3D. En particulier 5 volumes seront comparés ci-après (voir Figure 1).



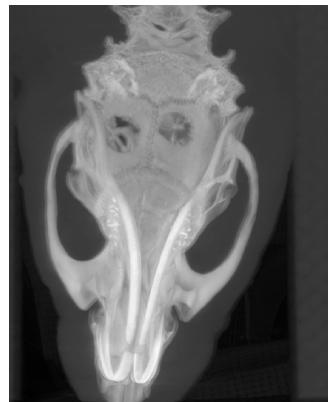
(a) IRM coeur, coupe axiale (8)



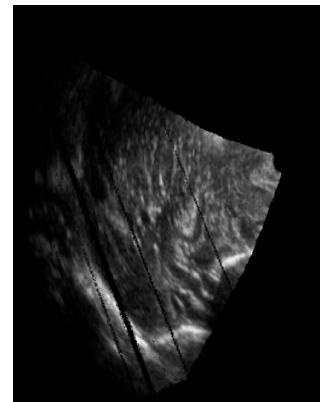
(b) IRM Flair coupe axiale (10)



(c) IRM T1 coupe coronnale (88)



(d) TDM coupe coronnale (max Projection)



(e) Ultrason coupe sagittale (233)

FIGURE 1 – Différentes coupes pour les images traitées question 2

2.1 Taille des voxels et des images

Pour retrouver les informations liées à la taille des images et la taille des voxels associés, la bibliothèque nibabel a été utilisée. Le tableau suivant fait état des résultats.

	TDM	IRM-flair	IRM-T1	IRM coeur	ultrason
Taille image	(384, 433, 532)	(208, 256, 22)	(192, 256, 176)	(256, 256, 14)	(323, 366, 371)
Taille voxels	(0.100, 0.100, 0.100)	(0.820, 0.820, 7.425)	(1.0, 1.0, 1.0)	(0.195, 0.195, 1.0)	(0.3, 0.3, 0.3)

Les valeurs de la section "taille de l'image" représentent, dans l'ordre : le nombre de pixels sur l'axe x, le nombre de pixels sur l'axe y, le nombre de couche sur l'axe z. La section "taille voxels" rend compte de la taille des voxels de la même façon. On remarque que les images TDM, IRM-T1 et ultrason possèdent des voxels isotropiques.

2.2 Contraste de Michelson et RMS

Deux mesures de contraste ont été effectuées sur les volumes entiers grâce au langage Python. La première ligne du tableau suivant fait état du résultat du calcul du contraste de Michelson sur chacun de nos volumes (pour rappel ce contraste correspond au ratio entre la plage d'intensité présente et les valeurs extrêmes). La deuxième ligne nous montre les résultats pour le calcul du contraste Root Mean Square (RMS) sur les volumes entiers (contraste considérant plutôt la taille de la dispersion autour de la moyenne).

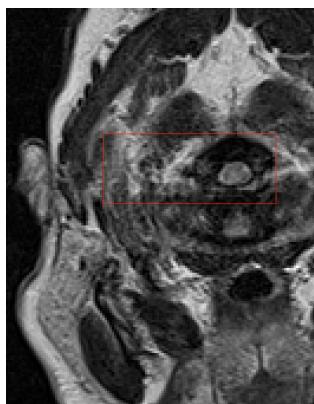
Nous notons cependant que ces calculs réalisés sur les volumes entiers ne sont pas forcément significatifs lors de nos visualisations 2D (et il serait judicieux comparer ces calculs 3D lorsqu'on les applique sur une coupe 2D en cours d'analyse).

	TDM	IRM-flair	IRM-T1	IRM coeur	ultrason
Michelson	3.178171	1.000000	1.000000	0.999980	1.000000
RMS	587.864899	191.872844	161.311491	0.023725	0.000373

2.3 Plus petite structure détectable et volume partiel

Dans les images, les plus petites structures détectables sont toujours au moins plus grande que la taille des voxels.

En utilisant le logiciel ImageJ, on semble observer des effets de volume partiel dans chacune des images à analyser dans cette partie. Cependant, rappelons que le volume partiel est lié au fait que certaines structures sont plus petites que la résolution de l'image, or, à cause de notre méconnaissance des structures analysées, il est parfois difficile de savoir si le phénomène est lié à un bruit (ou lié au contraste insuffisant) plutôt qu'à une structure plus petite.



(a) IRM Flair, coupe axiale (1)



(b) IRM T1 coupe axiale (36)



(c) TDM coupe axiale (430)

FIGURE 2 – Exemples de zones où l'on peut observer du volume partiel

Ainsi, grâce à ImageJ, nous avons délimité une section en rouge dans laquelle il nous semble voir du volume partiel. Dans les deux premières images de notre exemple, les structures semblent se confondre dû à cet effet. Concernant la troisième image, nous avons choisi de zoomer cette dernière afin de bien mettre en évidence ce volume partiel.

2.4 Bruit dans nos images

Pour trouver le bruit présent dans chaque image, nous avons choisi de calculer et d'analyser l'histogramme pour une zone donnée de notre volume (sur une coupe particulière). Cette zone correspond à un endroit où il n'y a aucune structure dans l'image, et on s'attendrait ainsi à avoir un histogramme dans lequel il n'y aurait qu'une unique classe représentée. Pour faciliter le choix de la zone, nous avons utilisé le logiciel ImageJ.

Pour l'image TDM, l'analyse semble montrer qu'un bruit gaussien est présent. Par ailleurs, il semble être uniformément réparti sur l'image. De plus, comme on peut le voir Figure 3a, certains artefacts (sorte de discontinuités formant des "lignes" qui semblent être anormales à notre oeil non spécialiste) sont détectables.

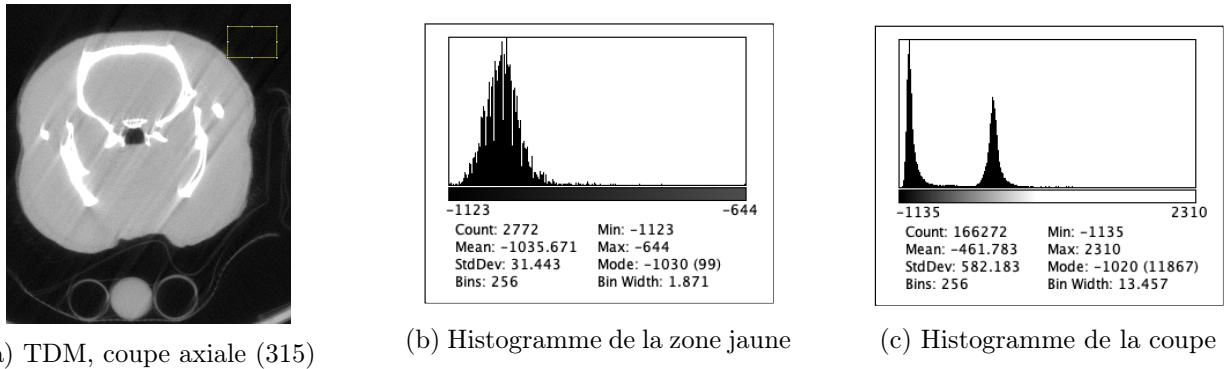


FIGURE 3 – Histogramme de certaines zones de l'image TDM - coupe axiale (315)

Pour l'image Flair, l'analyse semble montrer qu'un bruit de Rayleigh est présent. Cependant il ne semble pas vraiment être uniformément réparti sur l'image (plutôt au niveau des tissus acquise et peu sur le fond situé autour). Notons ici que notre analyse ne se base que sur l'édition de quelques histogrammes et qu'elle repose sur notre interprétation de la distribution estimée (cela pourrait donc également être un bruit gaussien).

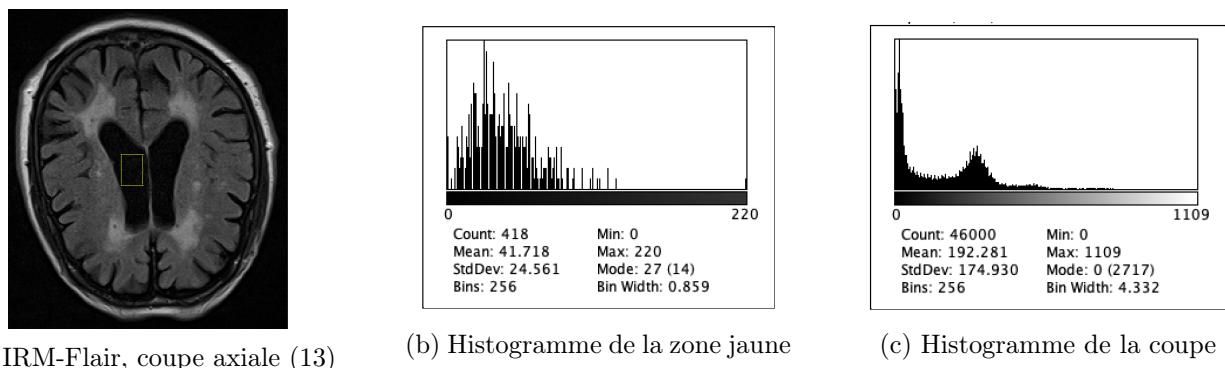


FIGURE 4 – Histogramme de certaines zones de l'image IRM-Flair - coupe axiale (13)

Pour l'image IRM-T1, l'analyse semble montrer qu'un bruit gaussien est présent. Par ailleurs, il semble être surtout réparti autour des tissus (et beaucoup moins présent dans le fond de l'image).

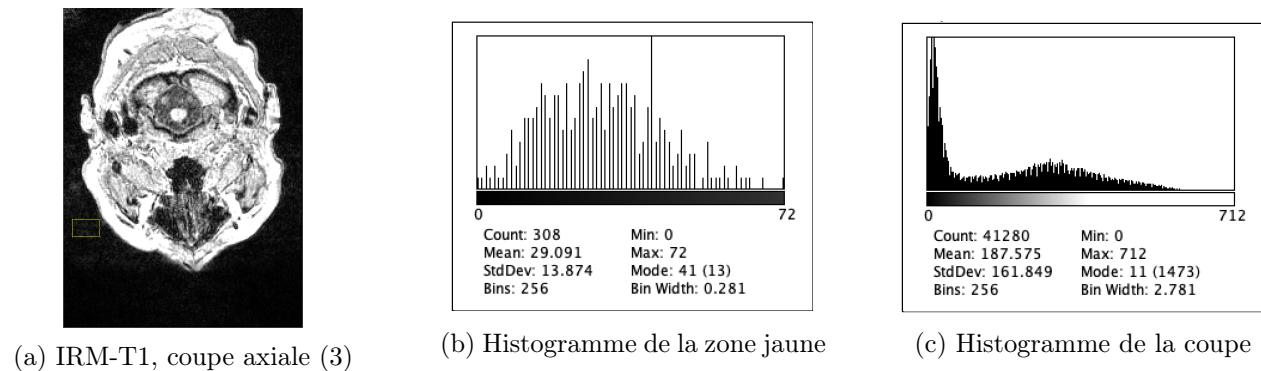
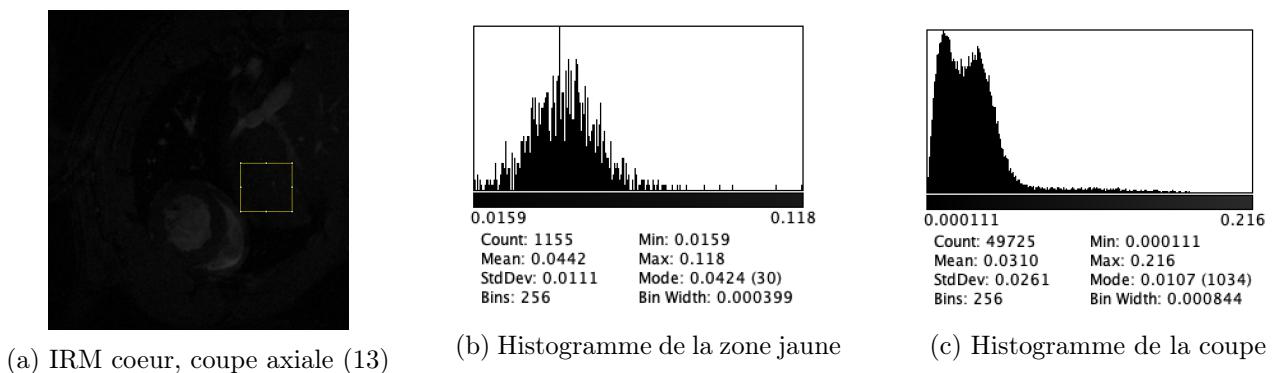


FIGURE 5 – Histogramme de certaines zones de l'image IRM-T1 - coupe axiale (3)

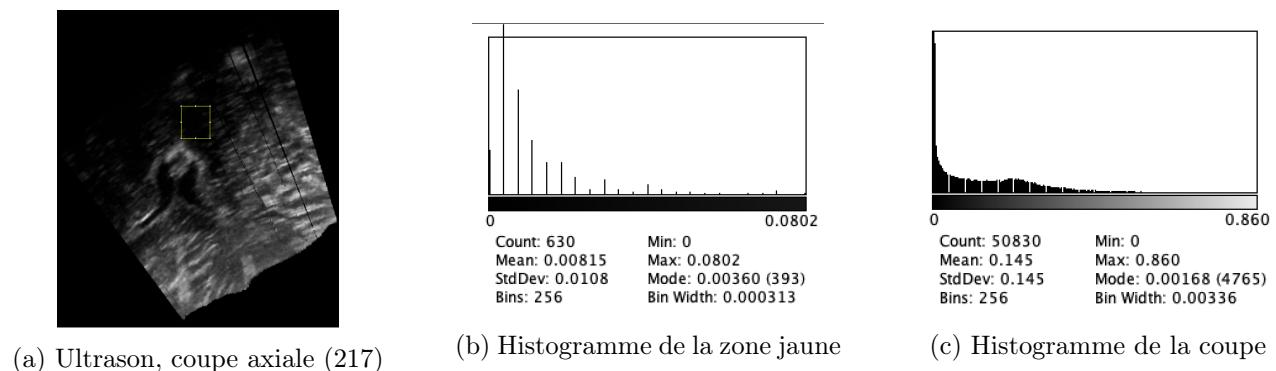
Pour l'image cœur, l'analyse semble montrer qu'un bruit gaussien est présent. Par ailleurs, il semble être uniformément réparti sur l'image.



(a) IRM cœur, coupe axiale (13) (b) Histogramme de la zone jaune (c) Histogramme de la coupe

FIGURE 6 – Histogramme de certaines zones de l'image cœur - coupe axiale (13)

Pour l'image ultrason, l'analyse semble montrer qu'un bruit gaussien est présent. Cependant, le peu d'information que l'on peut recueillir à l'aide de l'histogramme présenté et l'observation de l'image nous laisse penser que le bruit est peut être de nature poivre et sel ("speckle noise"). Par ailleurs, des artefacts sont présents dans l'image (voir figure 7a)



(a) Ultrason, coupe axiale (217)

(b) Histogramme de la zone jaune

(c) Histogramme de la coupe

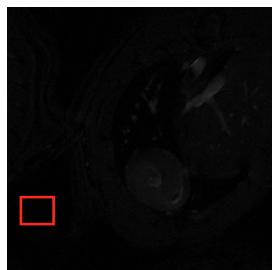
FIGURE 7 – Histogramme de certaines zones de l'image Ultrason - coupe axiale (217)

Il est important de noter qu'il n'est jamais possible d'obtenir une image sans bruit, ni même de connaître la nature exacte de ce bruit. Les résultats obtenus sont ainsi à considérer avec du recul.

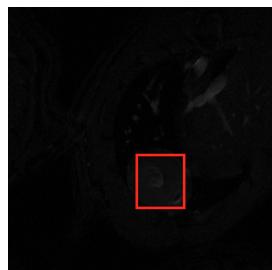
2.5 Calcul du SNR

Le calcul du SNR (Signal to noise ratio) pour "rapport signal sur bruit", correspond à une mesure de la qualité la transmission de l'information. Pour le calculer en pratique, on effectue le rapport entre la moyenne des intensités du signal et l'écart-type du fond. On suppose pour cela que le bruit présent sur l'image est gaussien.

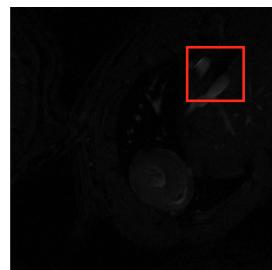
Dans ce devoir nous avons estimé que seules 3 images possédaient un bruit gaussien (TDM, coeur et T1). La difficulté du calcul à effectuer est que, dépendamment de la zone choisie, le calcul du SNR pourra différer. On propose donc pour chaque image le calcul du SNR pour deux signaux (jugés différents) par rapport à une zone de fond fixe. Les figures suivantes présentent les résultats. Globalement, nous avons essayé d'avoir une zone de fond sans signal (ie sans tissu ou autre élément d'intérêt) et de choisir des zones du signal "différentes".



(a) Zone de fond, std=0,00526



(b) Signal 1, moyenne=0,00742



(c) Signal 2, moyenne=0,0573

FIGURE 8 – En rouge les zones choisies pour le calcul du $\text{SNR}_1=14,10$ et $\text{SNR}_2=10,89$ de l'image IRM cœur



(a) Zone de fond, std=26,191



(b) Zone du signal 1, moyenne=1078,54



(c) Zone du signal 2, moyenne=988,242

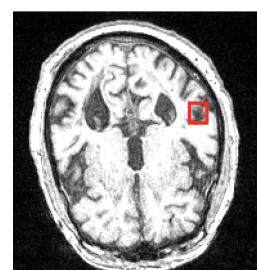
FIGURE 9 – Zones choisies pour le calcul du $\text{SNR}_1=41,17$ et $\text{SNR}_2=37,73$ de l'image TDM



(a) Zone de fond, std=18,747



(b) Zone du signal 1, moyenne=126,285



(c) Zone du signal 2, moyenne=204,795

FIGURE 10 – Zones choisies pour le calcul du $\text{SNR}_1=6,73$ et $\text{SNR}_2=10,92$ de l'image IRM T1

3 Question 3 : Méthodes de débruitage

Concernant le débruitage, nous avons considéré que les bruits présents dans ces images étaient de nature gaussienne et uniformément réparti dans l'image. Nous avons ainsi appliqué les filtres "médian", "nl means" et bilatéral" sur les images "flair" et "ultrason", et enfin affiché la différence des deux images afin de voir le bruit supprimé par le filtre. Pour cela, nous avons utilisé la bibliothèque "scikit-image" de Python. Le code est disponible en annexes.

3.1 Ultrason

La coupe de l'image d'ultrason utilisée pour le débruitage est la coupe sagittale 234. Cette image a été choisie pour ses dimensions d'origines ainsi que sa résolution. Concernant l'ultrason, on peut remarquer que les filtre n'apportent pas une différence remarquable à l'image de départ, bien que l'image filtrée par le bilatéral possède des contours plus lissés. Pour améliorer ce filtre, il aurait pu être intéressant de faire plusieurs itérations. L'analyse du bruit issu de la différence entre image filtrée et image d'origine indique que la méthode nl-mean semble assez performante : une forme de bruit uniformément réparti sur le tissu a été retiré (image 3).

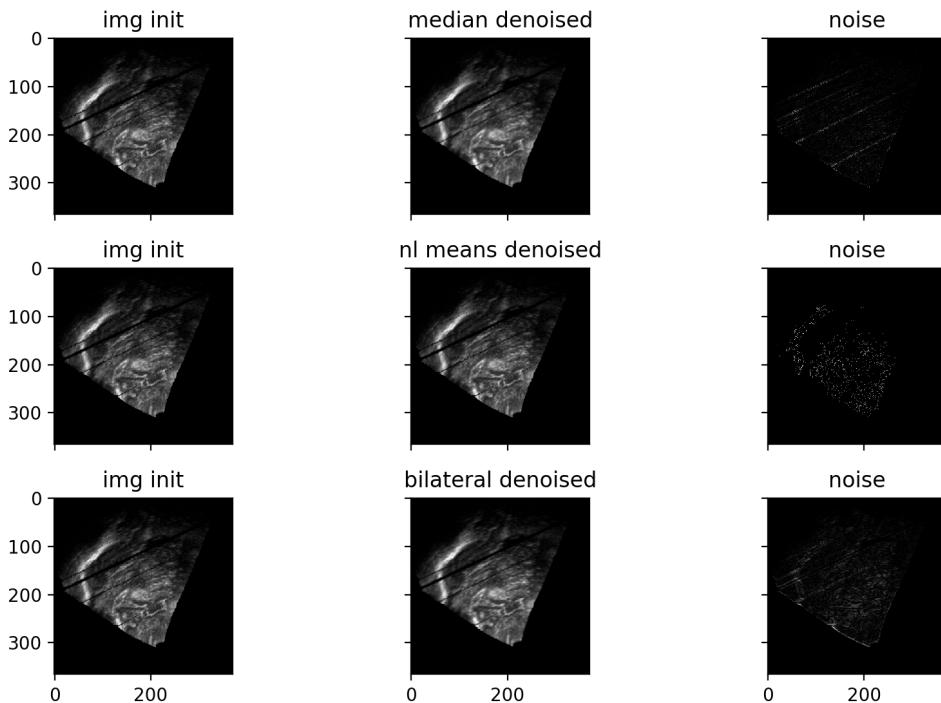


FIGURE 11 – Débruitage ultrason coupe sagittale (234)

3.2 IRM Flair

La coupe concernant l'image Flair est la coupe axiale 12. On remarque que les images, une fois filtrées, sont relativement plus différentes que les images d'origine, notamment pour le filtre "bilateral" (cela

s'explique notamment par le choix de paramètre du sigma spatial correspondant à l'écart type de la distance entre deux rangs). Si l'on observe bien le résultat de ce dernier, on remarque que le filtre a été très mauvais et qu'il n'a absolument pas filtré l'image. Le filtre médian quant à lui a pu supprimer un petit peu de bruit dans l'image, mais finalement le filtre le plus performant dans l'image reste la méthode "nl means" qui semble avoir retiré le bruit sur les parties de notre signal sans pour autant toucher aux contours.

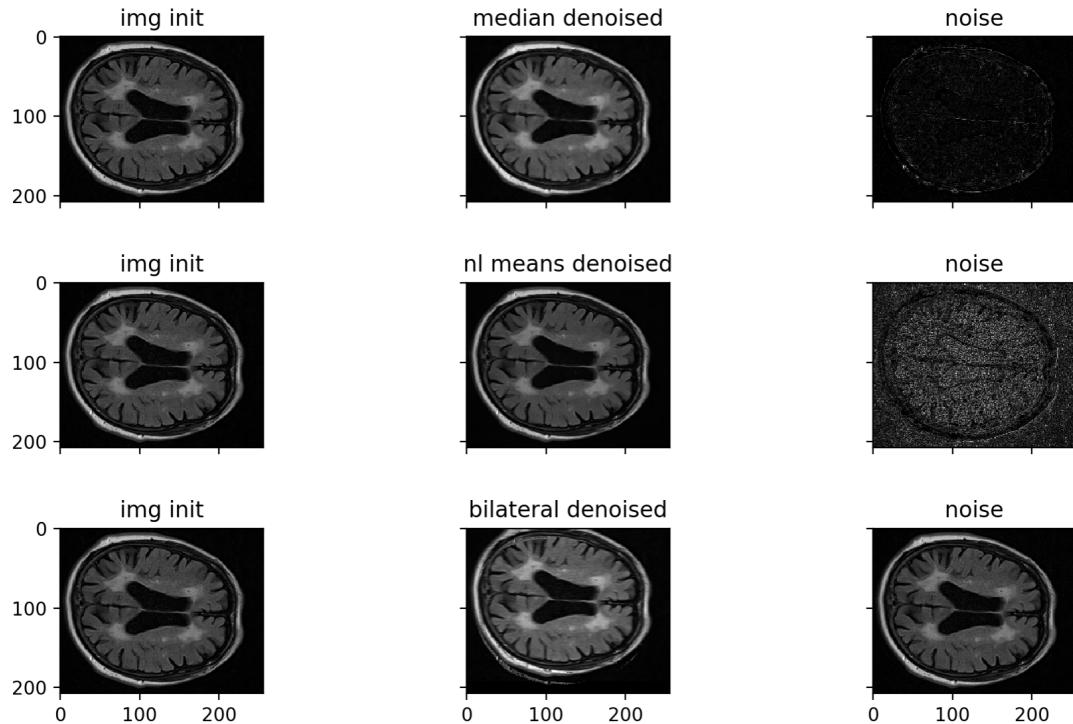


FIGURE 12 – Débruitage IRM Flair coupe axiale (12)

TD-reconstruction

October 6, 2019

1 TP1 : Imagerie Médicale - Annexe rapport

```
[1]: import numpy as np
import nibabel as nib
import matplotlib.pyplot as plt
import holoviews as hv
hv.extension('bokeh')
import panel.widgets as pnw
import panel as pn
import matplotlib.pyplot as plt
import param
import seaborn as sns
import scipy as sp
from scipy import stats
import pandas as pd
```

1.1 Question 1: Visualiser des volumes

1.1.1 “Faire une fonction viewer en Python permettant d'afficher l'ensemble des coupes dans un plan donné.”

```
[2]: def viewer(path):
    """
        Fonction prenant en entrée le chemin vers une image 3D et permettant
        →une visualisation
        selon 3 coupes possibles : 'axiale', 'coronale' et 'sagittale.
        path: string du chemin vers l'image
        :return: widget panel permettant la visualisation
    """
    if isinstance(path,str):
        image = np.squeeze(nib.load(path).get_data())
    else :
        image = np.squeeze(path)
    dict_label={'sagittal':0,'coronal':1,'axial':2}
```

```

    slider = pn.widgets.IntSlider(name='slider', value=0, start=0, end=image.
→shape[0])
    label_coupe = pn.widgets.Select(name='label_coupe', ↴
→options=['sagittal', 'coronal', 'axial'])

    def callback(target, event):
        target.end = image.shape[dict_label[event.new]]
        target.value = 0
        show_img(target.object, event.new)
    label_coupe.link(slider, callbacks={'value': callback})

    @pn.depends(slider.param.value, label_coupe.param.value)
    def show_img(index_image, label_coupe):
        if label_coupe == 'sagittal':
            bounds=(0,0,image.shape[1],image.shape[2])
            return hv.Image(np.rot90(image[index_image,:,:,:])), bounds=bounds).
→opts(colorbar=True,cmap='gray')
        elif label_coupe == 'coronal':
            bounds=(0,0,image.shape[0],image.shape[2])
            return hv.Image(np.rot90(image[:,index_image,:,:])),bounds=bounds).
→opts(colorbar=True,cmap='gray')
        else:
            bounds=(0,0,image.shape[0],image.shape[1])
            return hv.Image((np.rot90(image[:, :, index_image]))),bounds=bounds).
→opts(colorbar=True,cmap='gray')
    label_coupe.link(slider, callbacks={'value': callback})
    widgets = pn.Column("<br>\n# Image observée", slider, label_coupe)
    panel = pn.Row(widgets, show_img)
    %output size=200
    return panel

```

[3] : `def m_projection(path, axe, max_proj=True):`
 `"""`

Fonction permettant de renvoyant le maximum ou le minimum d'intensité ↴selon un axe défini

max_proj: booléen indiquand si l'on effectue une min ou une max ↴projection

path: string du chemin relatif vers l'image

axe: entier correspondant à l'axe sur lequel il faut trouver l'extremum

Pour rappel: 0 --> sagittal

1 --> coronal

2 --> axial

`"""`

image = np.squeeze(nib.load(path).get_data())

if max_proj:

 new_imgs = np.max(image, axis=axe)

```

    else:
        new_imgs = np.min(image, axis=axe)
    return hv.Image(np.rot90(new_imgs), bounds=[0,0,new_imgs.shape[0], new_imgs.
→shape[1]]).opts(colorbar=True,cmap='gray')

```

[4]: # Import des images que nous testerons dans la question 2:

```

imgs = {'TDM':'Data_MiseEnForme/CT/rat111.nii',
        'IRM_flair':'Data_MiseEnForme/IRM/Brain/flair.nii',
        'IRM_T1':'Data_MiseEnForme/IRM/Brain/t1.nii',
        'IRM_coeur':'Data_MiseEnForme/IRM/Heart/PetitAxe/Slice06.nii',
        'ultrason':'Data_MiseEnForme/Ultrasound/us.nii'}
# Petit test du viewer implémentée précédemment :
viewer(imgs['ultrason'])

```

[4]: Row

```

[0] Column
[0] Markdown(str)
[1] IntSlider(end=323, name='slider')
[2] Select(name='label_coupe', options=['sagittal', 'coronal', ...],
value='sagittal')
[1] ParamFunction(function)

```

[5]: m_projection(imgs['IRM_T1'], axe=2, max_proj=True)

[5]: :Image [x,y] (z)

1.2 Question 2 : Manipuler des images issues de différentes modalités d'acquisition

1.2.1 Question 2-a)

[6]: def get_sizes(header):
 """
 Fonction qui retourne la taille des voxels de l'image
 Ainsi que la taille de l'image
 """
 shape_img = header.get_data_shape()[:3]
 shape_voxel = header.get_zooms()[:3]
 return (shape_img, shape_voxel)

[7]: index=['shape_img', 'shape_voxel']
columns = []
shapes_imgs = []
shapes_voxels = []
for key, value in imgs.items():

```

columns.append(key)
_ = get_sizes(nib.load(value).header)
shapes_imgs.append(_[0])
shapes_voxels.append(_[1])
dataFrame = pd.DataFrame(data=[shapes_imgs, shapes_voxels], index=index, columns=columns)
dataFrame

```

pixdim[1,2,3] should be non-zero; setting 0 dims to 1

```
[7]:
```

	TDM	IRM_flair \	
shape_img	(384, 433, 532)	(208, 256, 22)	
shape_voxel	(0.100401, 0.100401, 0.100401)	(0.8203125, 0.8203125, 7.425003)	
	IRM_T1	IRM_coeur	ultrason
shape_img	(192, 256, 176)	(256, 256, 14)	(323, 366, 371)
shape_voxel	(1.0, 1.0, 1.0)	(0.1953125, 0.1953125, 1.0)	(0.3, 0.3, 0.3)

1.2.2 Question 2-b)

```
[8]: def michelson_calcul(img):
    """
    Fonction qui calcul le contraste de Michelson de l'image spécifiée
    """
    lmax = npamax(img)
    lmin = npamin(img)
    c_michelson = (lmax-lmin)/(lmax+lmin)
    return c_michelson

def rmsCalcul(img):
    """
    Fonction qui calcul le rms (root mean squared) de l'image spécifiée
    """
    img_moyenne = np.mean(img)
    img_centree = (img - img_moyenne) ** 2
    img_mean_centre = np.mean(img_centree)
    rms = np.sqrt(img_mean_centre)
    return rms
```

```
[9]: index=['michelson', 'RMS']
columns = []
michelson = []
rms = []
for key, value in imgs.items():
    columns.append(key)
    img = np.squeeze(nib.load(value).get_data())
```

```

# On choisit de découper le volume à sa moitié pour une coupe axiale:
michelson_img = michelson_calcul(img)
rms_img = rms_calcul(img)
michelson.append(michelson_img)
rms.append(rms_img)
dataFrame = pd.DataFrame(data=[michelson, rms], index=index, columns=columns)
dataFrame

```

pixdim[1,2,3] should be non-zero; setting 0 dims to 1

[9]:

	TDM	IRM_flair	IRM_T1	IRM_coeur	ultrason
michelson	3.178171	1.000000	1.000000	0.999980	1.000000
RMS	587.864899	191.872844	161.311491	0.023725	0.000373

1.3 Question 3 : Quelques techniques de débruitage

[11]:

```

from scipy.signal import medfilt
from skimage.restoration import denoise_nl_means, denoise_bilateral
from skimage.filters import gaussian

```

[73]:

```

# Pour effectuer le débruitage, on choisit une coupe pour notre étude:
# Ce sera la dixième de la coupe axial pour l'IRM Flair
img_flair= nib.load(imgs['IRM_flair']).get_data()[:, :, 12]
# Ce sera la 234 de la coupe sagittale
img_ultrason=nib.load(imgs['ultrason']).get_data()[234, :, :]

```

[116]:

```

def show_denoising_methods(img):
    %matplotlib notebook
    median_filtered = medfilt(img)
    nl_means_denoised = denoise_nl_means(img, patch_size=2, patch_distance=2,
                                          h=20)
    bilateral_denoised = denoise_bilateral(img, sigma_color=0, bins=2,
                                             multichannel=False, sigma_spatial=1)

    median_noise = np.abs(img - median_filtered)
    nl_noise = np.abs(img - nl_means_denoised)
    bilateral_noise = np.abs(img - bilateral_denoised)

    fig, axs = plt.subplots(3, 3)
    axs[0, 0].imshow(img, cmap='gray')
    axs[0, 0].set_title('img init')
    axs[0, 1].imshow(median_filtered, cmap='gray')
    axs[0, 1].set_title('median denoised')
    axs[0, 2].imshow(median_noise, cmap='gray')
    axs[0, 2].set_title('noise')

    axs[1, 0].imshow(nl_means_denoised, cmap='gray')
    axs[1, 0].set_title('nl means denoised')
    axs[1, 1].imshow(nl_noise, cmap='gray')
    axs[1, 1].set_title('nl noise')
    axs[1, 2].imshow(nl_means_denoised - nl_noise, cmap='gray')
    axs[1, 2].set_title('nl noise removed')

    axs[2, 0].imshow(bilateral_denoised, cmap='gray')
    axs[2, 0].set_title('bilateral denoised')
    axs[2, 1].imshow(bilateral_noise, cmap='gray')
    axs[2, 1].set_title('bilateral noise')
    axs[2, 2].imshow(bilateral_denoised - bilateral_noise, cmap='gray')
    axs[2, 2].set_title('bilateral noise removed')

```

```
axs[1, 0].imshow(img, cmap='gray')
axs[1, 0].set_title('img init')
axs[1, 1].imshow(nl_means_denoised, cmap='gray')
axs[1, 1].set_title('nl means denoised')
axs[1, 2].imshow(nl_noise, cmap='gray')
axs[1, 2].set_title('noise')

axs[2, 0].imshow(img, cmap='gray')
axs[2, 0].set_title('img init')
axs[2, 1].imshow(bilateral_denoised, cmap='gray')
axs[2, 1].set_title('bilateral denoised')
axs[2, 2].imshow(bilateral_noise, cmap='gray')
axs[2, 2].set_title('noise')
plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=1, hspace=None)
for ax in axs.flat:
    ax.label_outer()
show_denoising_methods(img_flair)
```

<IPython.core.display.Javascript object>

<IPython.core.display.HTML object>