# 3D Brain MRI feature extraction

Sagaf Youssouf

Université Paris-Est Créteil (UPEC)
International Master of Biometrics and Intelligent Vision
https://www.international-master-biometrics-intelligent-vision.org/

**Abstract.** Features extraction is the most crucial step in any image processing pipeline. This assertion is also true and relevant for 3D medical image processing. It allows the extraction of relevant information for medical image analysis. 3D brain MRI absorption is highest in dense tissue. Here you will see the method used to apply masks and filters for this type of medical image to extract features. The idea behind the technique performed is to exploit intensity patterns in order to take out a sub-region of an array. To demonstrate the effectiveness of our code, we visualize the result in each intermediate step in 3D using the script developed on the last assignment and in 2D dimensions. Because of its simplicity, our results suggest that the proposed feature extraction method can be applied in any 3D brain volume.

**Keywords:** Feature Extraction · 3D Brain MRI · Masks · Filters

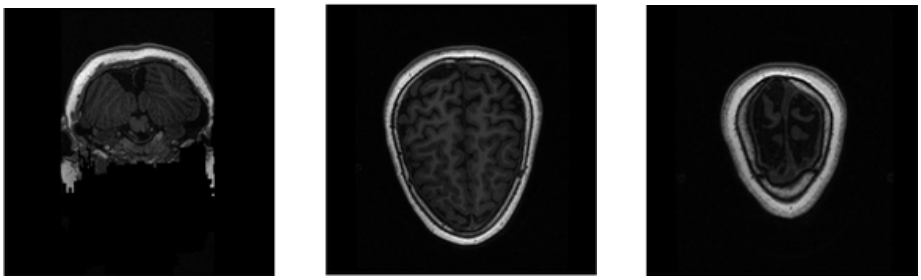
## 1 Introduction

In general, medical raw data such as MRI produced by medical technology are not suitable for analysis directly. This is due to the possible presence of noise in the data. On the other hand, not all data are important for analysis. For a 3D brain MRI, one may need to process only the brain tissue for different reasons. Therefore, it is necessary to consider only all relevant information from an image to be used for a given task. Feature extraction helps us to do that. Here we explore raw data and apply filter and mask in order to create a new representation of our input image. In the literature, there are many types of research that are focused on. The simplest one is to play with the intensity patterns with thresholding to create a mask and to control the data passing through the mask. In the next sections, we will explain the method applied to create a tissue mask and to detect the change of intensity from a 3D brain MRI.

## 2 Methods

**Image intensities exploration** : 3D volume elements are made up of a matrix of pixels and voxels. Here we are working on 3D brain MRI that have a very high absorption in dense tissue.

*Loading image* : To load the image, `load()` function from `nibabel` library is used by passing a T1-weighted brain .nii file. Afterwards, accessing to NumPy array is done using either `get_data()` or `get_fdata()` functions. The second one is highly recommended, because the first one is deprecated. We need to transpose the NumPy array using the `T` attribute for a good visualization. The shape of the 3D volume observed is 166x256x256. This means that we have 166 slices and each slice is a 2D array of 256x256. From the left to the right, grayscale plots of the $20th$, $120th$ and $140th$ slices are displayed below.



(a) $20th$ slice (b) $120th$ slice (c) $160th$ slice

Fig. 1: Grayscale plot of slices composing the 3D volume of brain MRI

*Checking intensities* : The datatype is given using `dtype` attribute in python and it controls the range of possible intensities. Here we have a datatype of `int16`, which can take integer values from $-32768$ to 32767. The minimum and maximum intensity values taken by the 3D volume are 0 and 10106 respectively. Other informations are given in figure 2.

```
print('[INFOS] Data type             :', im.dtype)
print('[INFOS] Min. intensity value  :', im.min())
print('[INFOS] Mean intensity value  :', np.round(im.mean(),2))
print('[INFOS] Max. intensity value  :', im.max())
print('[INFOS] Size of data          :', im.size)
print('[INFOS] Image data shape      :', im.shape)
```

```
[INFOS] Data type             : int16
[INFOS] Min. intensity value  : 0
[INFOS] Mean intensity value  : 535.65
[INFOS] Max. intensity value  : 10106
[INFOS] Size of data          : 10878976
[INFOS] Image data shape      : (166, 256, 256)
```

Fig. 2: Data information of the 3D brain image.

*Histogram and CDF* : Histogram displays the distribution of values of the 3D volume by binning each element by its intensity. Histogram is implemented in `scipy.ndimage` library. The following code illustrates how histogram and CDF are calculated. Plots of histogram and cumulative distribution function (CDF) are illustrated in figure 3. One can see that data clumped into two distributions consisting of brain tissues and non-brain tissues. We can separate these by setting a global threshold of 1200. Distribution is skewed toward low intensities so we may use equalization to redistribute value in order to optimize the full intensity range.

```
# Create a histogram, binned at each possible value
hist = ndi.measurements.histogram(im, min=0, max=10106, bins=10107)

# Create a cumulative distribution function
cdf = hist.cumsum()/hist.sum()
```

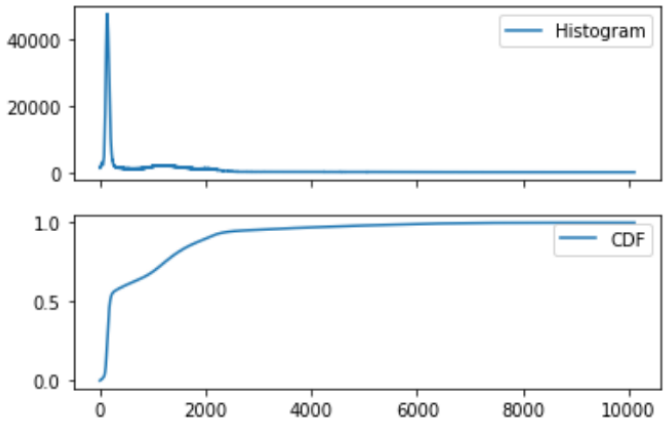



Fig. 3: Histogram (up) and CDF(bottom) curves of the 3D brain MRI.

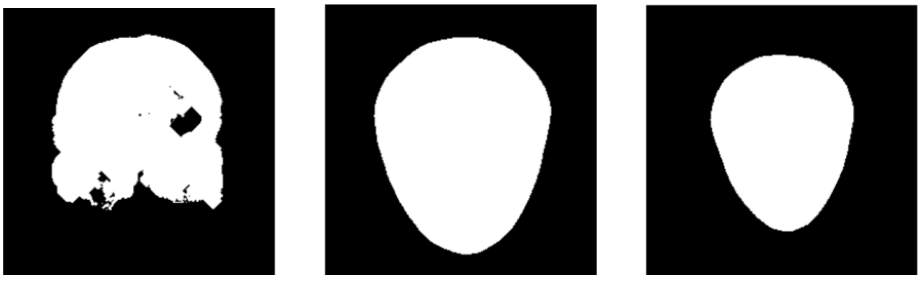
**Applying masks** : Here we want to create a tissue mask by removing all non-brain tissues. To do this, we applied a binary mask to the image to filter out pixels where the mask is `False`. To create the binay mask, we used this line of code : `maskHead = im>1200`. The result of somes of the created 2D binay masks are given in figure 5.
To apply the mask to the 3D image, NumPy's `where()` function is called as illustrated in the following code. This operation is followed by some morphological operations to tune imperfections. The result of this operation is a 3D head mask as illustrated in figure 5. Histogram and CDF of the computed 3D mask

(a) 20*th* slice (b) 120*th* slice (c) 160*th* slice

Fig. 4: Plots of 2D binay masks composing the 3D binary volume mask.

of the head are given in figure 6. From the histogram, one can see that intensity is equal to 0 from 0 to 1200 in the x-axis.

```
imHead = np.where(maskHead, im, 0)
imHead = ndi.binary_dilation(imHead, iterations=8)
imHead = ndi.binary_closing(imHead, iterations=8)
viewPlot(imHead)
```



(a) 20*th* slice (b) 120*th* slice (c) 160*th* slice

Fig. 5: Plots of the head masks

**Applying filters** : Now we need to use filters to transform the 3D volume based on values of the pixel in its surrounding. To smooth the image, we used a `3x3` kernel of `0.11` in each element as illustrated in the following portion of code in order to perform mean filtering. The kernel is convolved with the 3D brain MRI using `ndi.convolve()` function. The obtained result image is given in figure 7.

```
w = 0.11 * np.ones([3, 3, 3])
imFilt = ndi.convolve(im, w)
```

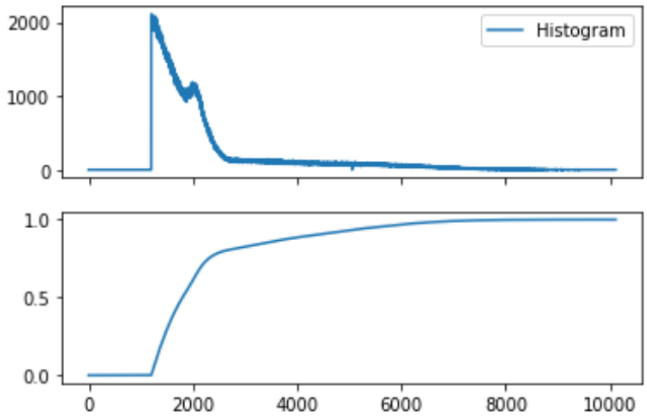
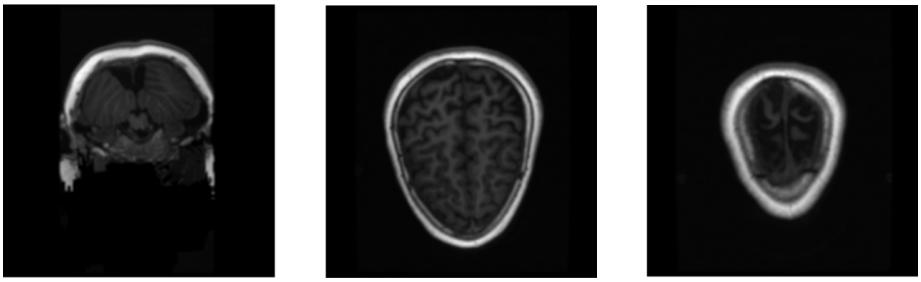



Fig. 6: Histogram (up) and CDF(bottom) curve of the 3D head mask.



(a) 20*th* slice (b) 120*th* slice (c) 160*th* slice

Fig. 7: Grayscale plot of slices composing the smoothed 3D volume of brain MRI

Now, filters are used for edge detection to detect the change of intensity from left to right. The kernel used is composed of `-1`, `0` and `1` values only as illustrated in the following code. The resulted image is given in figure 8 for one slice. We used `seismic` value of the `cmap` to display edges because using `gray` value does not highlight edges.

```
k = np.zeros([3, 3, 3])
k[0] = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])
k[1] = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])
k[2] = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]])

imEdge = ndi.convolve(im, k)
```

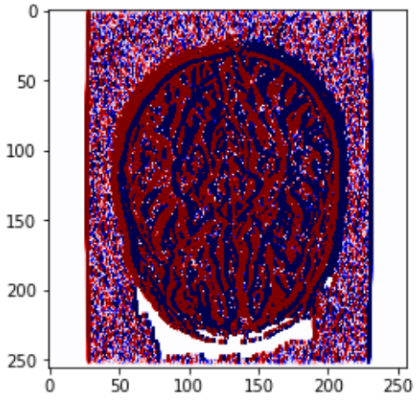
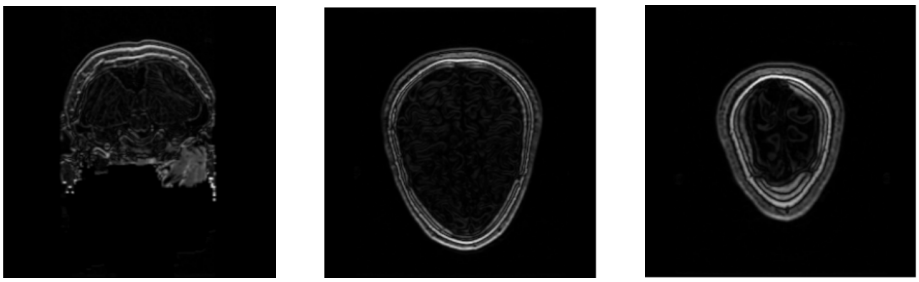



Fig. 8: Detected edges for the slice 60

Another popular edge detector is the Sobel filter. This filter provides extra weight to the center pixels of the detector. Here Sobel filtering is performed using `ndi.sobel` function to create the weights before convolving with the 3D brain MRI. Following images are the results for $20th$, $120th$ and 160 $th$ slices.



(a) $20th$ slice (b) $120th$ slice (c) $160th$ slice

Fig. 9: Grayscale plot of slices composing the 3D volume edges

## 3 Conclusion

In this lab, we exploit intensity patterns to select a sub-region of an array for feature extraction. We used convolutional filters to detect interesting features such as the edges. SciPy's ndimage module has been the main package used to process our brain MRI image. It contains a treasure trove of multidimensional image processing tools. This lab helps us to highlight skills in one of the most important steps in medical image analysis which is feature extraction.

## References

1. Tutorial on 3D medical image preprocessing, https://www.kaggle.com/akh64bit/full-preprocessing-tutorial
2. Masks and Filters in Biomedical Image Analysis, https://goodboychan.github.io/chans_jupyter/python/datacamp/vision/2020/08/15/01-Masks-and-Filters-in-Biomedical-Image-Analysis.html.
3. Medical Images In python (Computed Tomography)s, https://vincentblog.xyz/posts/medical-images-in-python-computed-tomography.
4. Chowdhary, C. L. and D. Acharjya. "Segmentation and Feature Extraction in Medical Imaging: A Systematic Review." Procedia Computer Science 167 (2020): 26-36.
5. Jude Hemanth D., Anitha J. (2012) Image Pre-processing and Feature Extraction Techniques for Magnetic Resonance Brain Image Analysis. In: Kim T., Ko D., Vasilakos T., Stoica A., Abawajy J. (eds) Computer Applications for Communication, Networking, and Digital Contents. FGCN 2012.