CS2204
Assignment No. 4

Purpose:
The goal of this assignment is to give you more experience with using NumPy in an actual science domain.

Introduction:
In this assignment, we will further study NumPy arrays[1] by using them in a practical application: contrast enhancement of medical images. Specifically, we will deal with magnetic resonance (MR) images. Magnetic resonance imaging is a technique that combines a strong magnetic field with radio frequency pulses to produce a three-dimensional image of whatever is being scanned. In the medical field, MR imaging provides very good contrast between different soft tissues in the body.

Even so, the raw MR data is often processed to make it easier to understand by someone examining it. One of the most common types of processing is contrast enhancement. In this assignment, you will write a program that will enhance the contrast of an MR volume using a particular algorithm called *histogram equalization*. The histogram of an image is a graph of the number of pixels (or *voxels*, when discussing 3D images) with a given intensity value. Often the histogram of an image has intensities that are clumped together. Histogram equalization spreads these intensities out in a particular, optimal way that results in a contrast-enhanced image.

Functional Specifications:

1. Read in the MR volume
The first step in the process is to read in the MR data. The MR data is stored in a particular file format called NIfTI[2], and you've been provided a Python library that will read in this file format. The code to read this image has also been provided. For those who would like to get a better idea of the image they're working with, download mricron and use it to open the provided NIfTI. Note that the image provided has dimensions 170x256x256. When displaying the image, you will be plotting the 85th lateral slice (side to side), which in the array is [85, : , :].

2. Normalize the Data
Usually, when we display an image, the intensity in a color channel can have 256 levels (not always, but most display devices, including your computers, support this). MR images don't have any color so they're represented as grayscale images, and when we display them, we want them to have 256 intensity levels. However, the technology of acquiring MR data gives you numbers that are floating point and greatly exceed the range from 0 to 255, so the first thing we would like to do in our processing is *normalize* the data so that every voxel is an integer between 0 and 255, inclusive. This step is not strictly necessary but makes later steps easier. Let i be the intensity of a voxel and i_n be the new intensity after normalization. The formula for normalization is:

$$i_n = round((\frac{i - i_{min}}{i_{max} - i_{min}} - i_{min}) \times 255)$$

where i_{max} and i_{min} are the maximum and minimum intensities over the entire volume.

---

[1] You may find the flat iterator of an ndarray useful in this assignment.

[2] http://nifti.nimh.nih.gov/nifti-1

3. Compute the Histogram

For a digital image, the histogram *H* can be thought of as an array. The index of the array is the intensity. The value of the array at that index, *H[i]*, is the number of pixels or voxels having that intensity. For our normalized MR volume, the histogram array will range from 0 to 255. You need to compute the histogram of the MR volume. This operation will likely take a few minutes.

4. Compute the Cumulative Distribution Function

The cumulative distribution function (CDF) is what we are going to compute the histogram equalization with. It contains exactly the same information as the histogram, but it represents it in a different way. Thus, the CDF is an array like the histogram, but the value at a particular index is the number of voxels in the MR volume of that intensity or less. Thus:

$$CDF[i] = \sum_{j=0}^{i} H[j]$$

You should compute the CDF for the MR volume. Note that CDF[255] should equal the number of voxels in the image.

5. Equalize the Histogram

In histogram equalization, we're trying to define a new image based on the old image that is as flat as possible. If a histogram is flat, then the cumulative distribution function will be a line or a ramp. Now that we've computed the histogram and the cumulative distribution function, we need to define a transformation that will flatten them out, and that will give us a rule for assigning intensities to voxels in the new image. It turns out that the rule for this looks quite similar to our normalization equation above:

$$I_n(v) = round((\frac{CDF[I(v)] - CDF_{min}}{N - CDF_{min}} \times 255)$$

where *v* is a particular voxel, *I* is the original image (normalized), I_n is the new image, CDF_{min} is the minimum nonzero value CDF of the original image, and *N* is the total number of voxels in the image.

Compute the histogram equalization of the MR volume.

6. Plot Results

As an example, Figure 1 shows an original slice from an MR volume and Figure 2 shows the histogram equalized slice. Figure 3 shows the before and after histograms, and Figure 4 shows the before and after cumulative distribution functions. As mentioned before, the code to display these images has been provided, you just need to replace the placeholders with your variables names. However, it would not be a horrible idea to use the variable names provided for your own code.

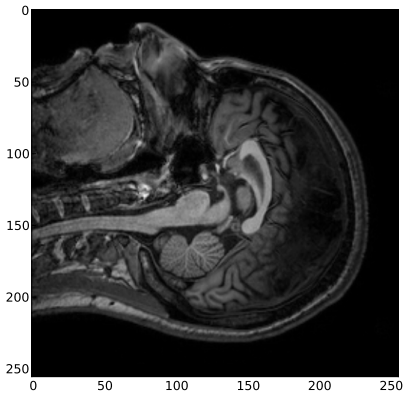


Figure 1 Before

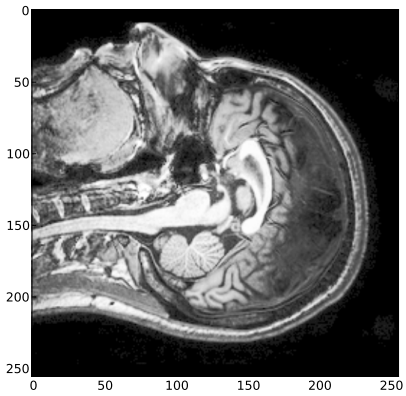
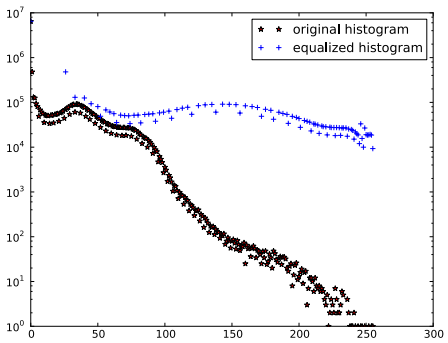
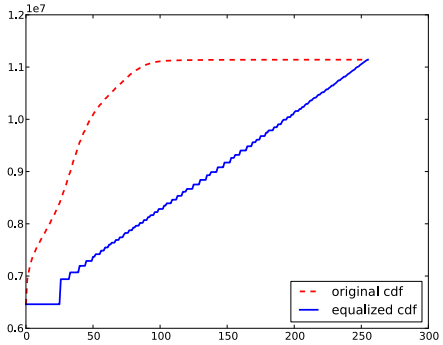


Figure 2 After





Requirements:
You should display the middle slice (85) from the original image as a point of reference. After the histogram-equalized image has been computed, display the middle slice from this new image. Compute a new histogram and CDF for the new image. Display both histograms together on one graph. Then display both CDFs together on a second graph. The code for plotting graphs and displaying images has been provided.

Submission for grading:
When you have completed your work on this assignment, please submit the **project4.py** source file for grading along with the 4 images. You submit the files by visiting the assignment page in Blackboard and attaching the files (one at a time) by clicking on the "Browse my computer" button and finding the file to attach.

Grading:
This project is worth 50 points. Your grade on this project will be based on the following:

1. The correctness of your methods implemented in the project4.py file.
2. The use of built-in Python capabilities (list comprehensions, array sections, etc.).
3. The use of good programming style.

You should also review the syllabus regarding the penalties for late programming assignments.

Further Reading:

1. Image processing, histogram equalization: *Digital Image Processing*, 3rd ed., R. C. Gonzalez and R. E. Woods, Prentice-Hall, 2008.
2. Magnetic Resonance Imaging: http://www.cis.rit.edu/htbooks/mri/ and https://masi.vuse.vanderbilt.edu/index.php/MRI_Introduction.