Enhancing Medical Images using Gamma Correction and Histogram Equalization

Your Name

March 28, 2023

Medical images play a vital role in the diagnosis and treatment of various diseases. However, these images are often low contrast, noisy, and require enhancement to visualize the relevant features. In here, we will discuss how to enhance medical images using gamma correction and histogram equalization in Python.

1 Gamma Correction

Gamma correction is a non-linear image enhancement technique used to adjust the brightness and contrast of images. It involves raising the pixel values of an image to a power (gamma) to increase or decrease the brightness of the image. Gamma correction is particularly useful for enhancing images that are too dark or too bright.

To implement gamma correction in Python, we use the `numpy.power` function. The code snippet below demonstrates how to perform gamma correction on an input image:

```
import numpy as np
import matplotlib.pyplot as plt

Load the input image

img = plt.imread('input_image.png')

Perform image enhancement using gamma correction

gamma = 1.5
img_enhanced = np.power(img, gamma)
```

In the code above, we first load the input image using the `matplotlib.pyplot.imread` function. We then define a gamma value of 1.5 and use the `numpy.power` function to raise the pixel values of the image to the power of gamma, thereby enhancing the image.

2 Histogram Equalization

Histogram equalization is another image enhancement technique used to adjust the contrast of images. It works by redistributing the pixel intensities in an image such that the histogram of the output image is more uniformly distributed. Histogram equalization is particularly useful for enhancing images with low contrast.

To implement histogram equalization in Python, we first convert the input image to grayscale using the dot product of the RGB values with a set of weights. We then normalize the grayscale image to have values between 0 and 255 and apply the `numpy.histogram` and `numpy.cumsum` functions to compute the cumulative distribution function (CDF) of the pixel intensities. Finally, we use the CDF to map the pixel values of the input image to new values in the output image.

The code snippet below demonstrates how to perform histogram equalization on an input image:

```
import numpy as np
import matplotlib.pyplot as plt

Load the input image

img = plt.imread('input_image.png')

Convert the image to grayscale

img_gray = np.dot(img[..., :3], [0.299, 0.587, 0.114])

Normalize the grayscale image

img_norm = (img_gray - img_gray.min()) / (img_gray.max() - img_gray.min())

Compute the histogram and CDF of the normalized image

hist, bins = np.histogram(img_norm.flatten(), 256, [0, 1])
cdf = hist.cumsum()
cdf_normalized = cdf / cdf.max()

Map the pixel values of the input image to new values in the output image using the CDF

img_equalized = np.interp(img_norm.flatten(), bins[:-1], cdf_normalized) * 255
img_equalized = img_equalized.reshape(img_norm.shape).astype(np.uint8)
```

In the code above, we first load the input image using the `matplotlib.pyplot.imread` function. We then convert the image to grayscale using the dot