k-space_wght_msk_for_MRI_denoising

k-space weighting and masking for denoising of MRI image without blurring or losing contrast, as well as for brightening of the objects in the image with simultaneous noise reduction (on the example of Agilent FID data).

1 Introduction

Raw MRI (Magnetic Resonance Imaging) data is stored in k-space. Each line of k-space represents a nuclear magnetic resonance (NMR) echo signal with the signal maximum located in the middle of the line. There is no NMR signal before the echo begins nor after the echo ends. The center of k-space hosts the strongest signals - the signal drops not only to the sides but generally towards the k-space edges (this is natural result of the physical nature of frequency and phase encoding in MRI experiment).

It is a fair assumption, that beyond some distance from the k-space center there is no usefull NMR data - just electric noise. Removing that 'empty' region results in excluding some noise from the resulting MRI image. It is worth remembering that electric noise covers all the k-space, so this technique would not remove all the noise.

To increase the separation between signal and noise, each data point can be multiplied by a value depending on its magnitude. This way the stronger the signal the more it is amplified. Strongest signals are located in the centre of k-space, so this amplification technique will brighten the object on the resulting MRI image due to change of balance between the magnitude of information about objects coloring and other image data. Here is why.

Every point in k-space represents a wave in the reconstructed image - a different spatial frequency at a different angle. The center of the k-space hosts low spatial frequencies (these data points in k-space give only large waves on the image - information about large areas of bright and dark, i.e. contrast information). The farther from the center the higher spatial frequency, so edges and detail information is located on the periphery of k-space.

To restore some contrast, the k-space datapoints with greatest values have to be scaled down adequately.

This software utilises both of the noise reduction strategies described above and has a contrast restoration option implemented.

2 The sample data

The data was acquired in the frame of the project described in the article titled *Alcohol-fixed specimens for high-contrast post-mortem MRI* (https://doi.org/10.1016/j.fri.2021.200449). The specimen is one of the untreated mice.

MRI was performed at 17°C on a 9.4 T horizontal bore Agilent MRI scanner using a 40 mm diameter Agilent millipede coil for transmission and reception. The experiment was multi spin echo multi slice imaging (TR = 8s, TE = 7 ms, NE = 32). There were 12 spatial slices, each repeated for 32 different echo time values, resulting in total slice number of 384. Slice dimensions were 40 mm x 40 mm with slice thickness of 1 mm and data matrix size 128 x 128.

3 The software

For the full code please see 'wght_msk_kspace.py' script file. The script contains 2 functions: 'wght_msk_kspace' for all the calculations and 'main' which provides input parameters for 'wght_msk_kspace' function, runs it and retrieves the results. It also creates global variables for the results to be available after the run completion.

3.1 Necessary python libraries:

- nmrglue,
- numpy,
- matplotlib,
- cv2.

3.2 Input parameters for 'wght_msk_kspace' function:

1. .fid folder location: **path** (string variable, e.g. "D:/mems_20190406_02.fid" or just "mems_20190406_02.fid" if located in one folder with the script),
2. total number of slices in the imaging experiment: **number_of_slices** (integer, e.g. 10)
3. selected slice number: **picked_slice** (integer, e.g. 5),
4. exponent in the signal weighting equation: **weight_power** (floating point number, e.g. 0.09),
5. restoring contrast option: **contrast** (boolean value: 1 for denoising including contrast restoration, 0 for brightening of the objects in the image with simultaneous noise reduction).

3.3 How 'wght_msk_kspace' function works

The function starts with importing data from .fid folder specified in **path**. Single .fid file contains raw data for the whole imaging experiment. The function needs **number_of_slices** to downsample the data to extracts k-space corresponding to the **picked_slice**.

If **contrast** is set to 0, each element of the k-space is multiplied by its absolute value to the power of **weight_power** value.

If **contrast** is set to 1, each element of the k-space is multiplied by its absolute value to the power of **weight_power** value times 3. Subsequently, contrast is restored by finding k-space values greater than absolute value of the maximal k-space value devided by 6. Those big values are divided by their absolute value to the power of **weight_power** value.

Next, all the k-space values outside the circle inscribed in the k-space are changed to 0 and the k-space gets normalised to its original maximal value.

In the next step, MRI images are reconstructed, both from original k-space and modified k-space, using 2-dimensional Fast Fourier Transform (2D FFT).

The results get sumarised in an illustration showing: the original k-space, the modified k-space, the original MRI image and the denoised/brightened MRI image. In the last step, the function returns the results in a form of 4 data arrays of complex numbers.

3.4 'wght_msk_kspace' function definition

In [1]:

```
import nmrglue as ng
import numpy as np
import matplotlib.pyplot as plt
import cv2

def wght_msk_kspace(path, number_of_slices, picked_slice, weight_power, contrast):
    """
    k-space weighting and masking for denoising of MRI image without blurring or losing contrast,
    as well as for brightening of the objects in the image with simultaneous noise reduction
    (for Agilent FID data).
    Input:
        .fid folder location: path [str],
        total number of slices in the MRI experiment: number_of_slices [int],
        selected slice number: picked_slice [int],
        exponent in the signal weighting equation: weight_power[float]
        restoring contrast: contrast [bool] (1 for denoising, 0 for brightening).
    """

    # import k-space data
    echoes = ng.agilent.read(dir=path)[1]
    kspace = echoes[picked_slice - 1 : echoes.shape[0] : number_of_slices, :]  # downsampling to one slice
    del path, echoes, number_of_slices, picked_slice

    # k-space weighting
    if contrast:
        kspace_weighted = kspace * np.power(abs(kspace), 3*weight_power)
        # contrasting
        a = kspace_weighted[abs(kspace_weighted) > abs(np.max(kspace_weighted))/6]
        a = a / np.power(abs(a), weight_power)
        kspace_weighted[abs(kspace_weighted) > abs(np.max(kspace_weighted))/6] = a
        title = 'Denoised image'
        del a
    else:
        kspace_weighted = kspace * np.power(abs(kspace), weight_power)
        title = 'Brightened image'
    del contrast, weight_power

    # k-space masking
    r = int(kspace.shape[0]/2)
    mask = np.zeros(shape=kspace.shape)
    cv2.circle(img=mask, center=(r,r), radius = r, color =(1,0,0), thickness=-1)
    kspace_weighted = np.multiply(kspace_weighted, mask)
    del mask, r

    # normalization
    kspace_weighted = kspace_weighted / (np.max(abs(kspace_weighted)) / np.max(abs(kspace)))

    # reconstructing the original image
    ft1 = np.fft.fft2(kspace)                 # 2D FFT
    ft1 = np.fft.fftshift(ft1)                # fixing problem with corner being center of the image
    ft1 = np.transpose(np.flip(ft1, (1,0)))   # matching geometry with VnmrJ-calculated image (still a bit shifted)

    # reconstructing denoised image
    ft2 = np.fft.fft2(kspace_weighted)        # 2D FFT
    ft2 = np.fft.fftshift(ft2)                # fixing problem with corner being center of the image
    ft2 = np.transpose(np.flip(ft2, (1,0)))   # matching geometry with VnmrJ-calculated image (still a bit shifted)

    # visualization
    plt.rcParams['figure.dpi'] = 600
    plt.subplot(141)
    plt.title('Original k-space', fontdict = {'fontsize' : 7}), plt.axis('off')
    plt.imshow(abs(kspace), cmap=plt.get_cmap('gray'), vmax=int(np.mean(abs(kspace))*7))
    plt.subplot(142)
    plt.title('Modified k-space', fontdict = {'fontsize' : 7}), plt.axis('off')
    plt.imshow(abs(kspace_weighted), cmap=plt.get_cmap('gray'), vmax=int(np.mean(abs(kspace))*7))
    plt.subplot(143)
    plt.title('Original image', fontdict = {'fontsize' : 7}), plt.axis('off')
    plt.imshow(abs(ft1), cmap=plt.get_cmap('gray'))
    plt.subplot(144)
    plt.title(title, fontdict = {'fontsize' : 7}), plt.axis('off')
    plt.imshow(abs(ft2), cmap=plt.get_cmap('gray'))
    plt.tight_layout(pad=0, w_pad=0.2, h_pad=0)
    plt.show()
    del title

    # return data
    return kspace, kspace_weighted, ft1, ft2
```

4 Sample results

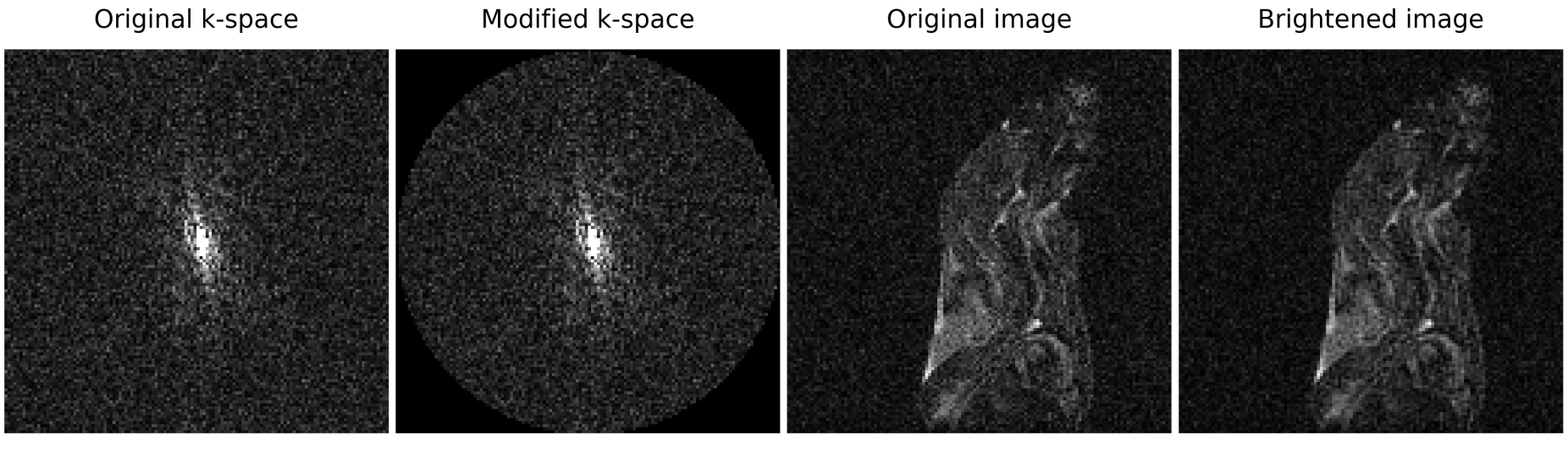
The easiest way to find optimal **weight_power** value for denoising without brightening is to start without restoring contrast (**contrast** = 0). The right value will be the one resulting in an image with minimal noise without much image brightenning. The **contrast** variable can then be set to 1.

The right value of **weight_power** for brightening of the objects in the image depends on individual expectations.

4.1 weight_power value impact (for contrast = 0)

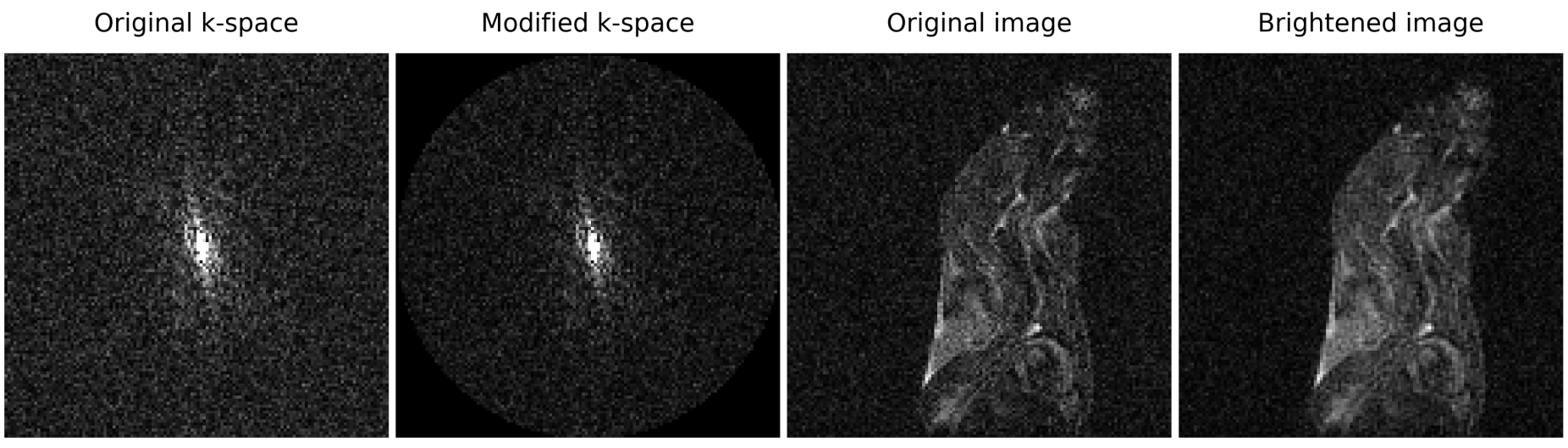
- Too small weight_power value - SNR (signal to noise ratio) is below the software capability.

In [3]:
```
wght_msk_kspace('mems_20190406_02.fid', 384, 119, 0.001, 0);
```



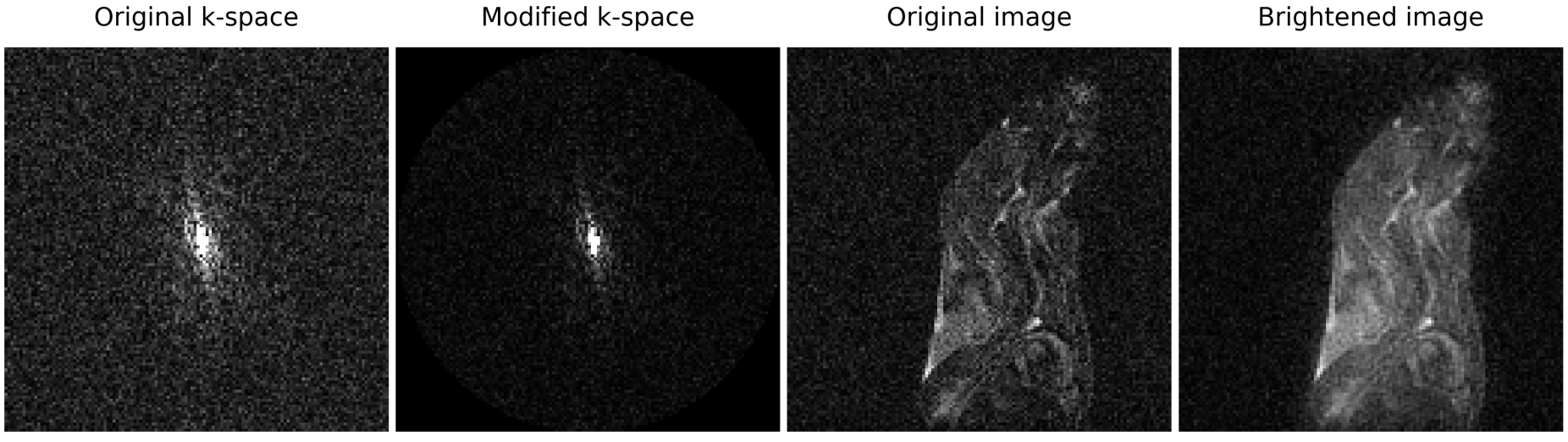
- weight_power value optimal for denoising without brightening.

In [4]:
```
wght_msk_kspace('mems_20190406_02.fid', 384, 119, 0.09, 0);
```



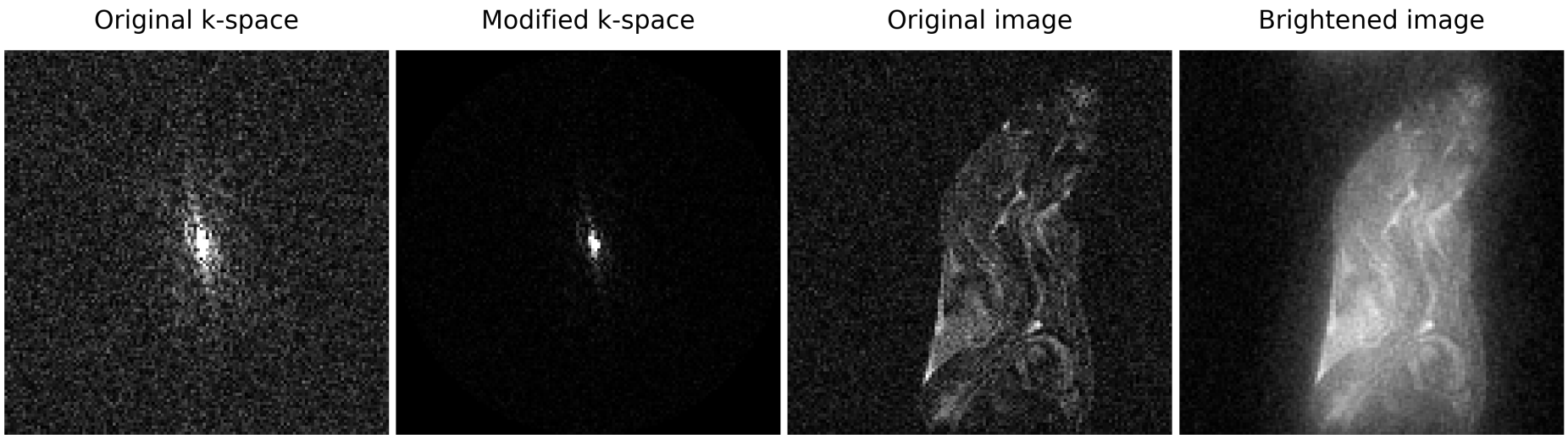
- An example of weight_power value for brightening of the objects in the image.

In [5]:
```
wght_msk_kspace('mems_20190406_02.fid', 384, 119, 0.23, 0);
```



- Too large weight_power value.

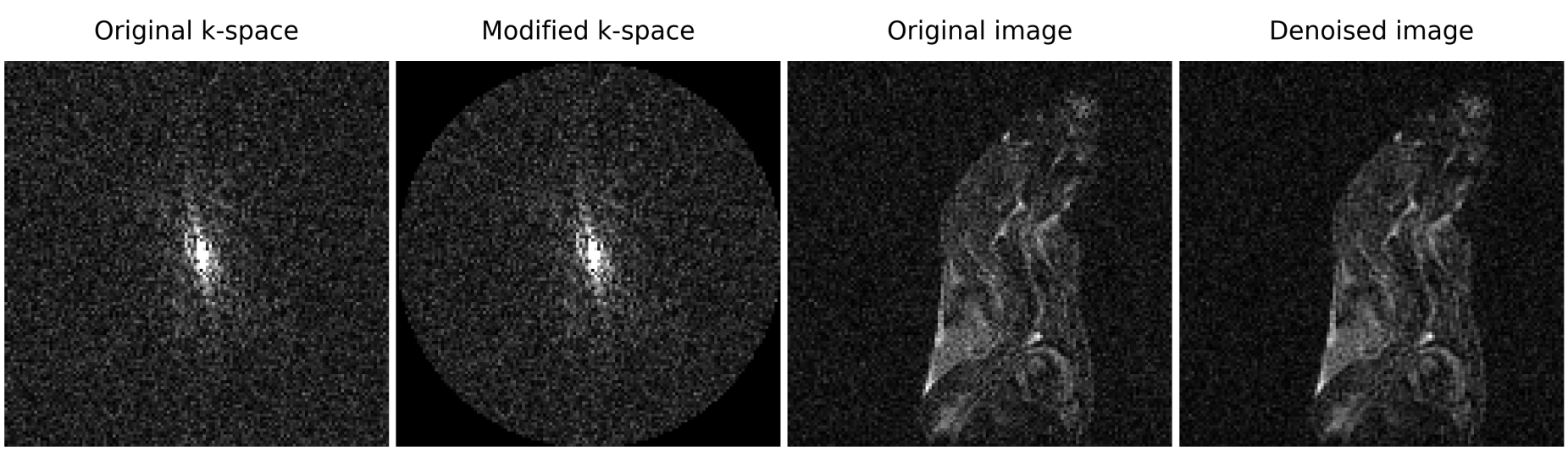
In [6]:
```
wght_msk_kspace('mems_20190406_02.fid', 384, 119, 0.5, 0);
```



4.2 weight_power value impact (for contrast = 1)

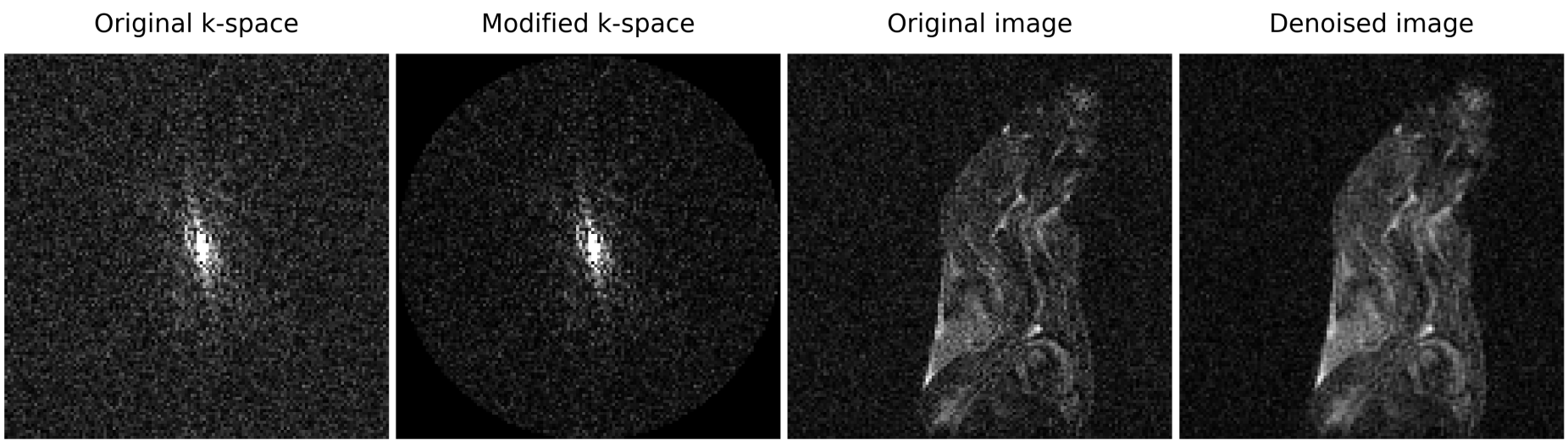
- Too small weight_power value - SNR (signal to noise ratio) in the image is below the software capability.

In [7]:
```
wght_msk_kspace('mems_20190406_02.fid', 384, 119, 0.001, 1);
```



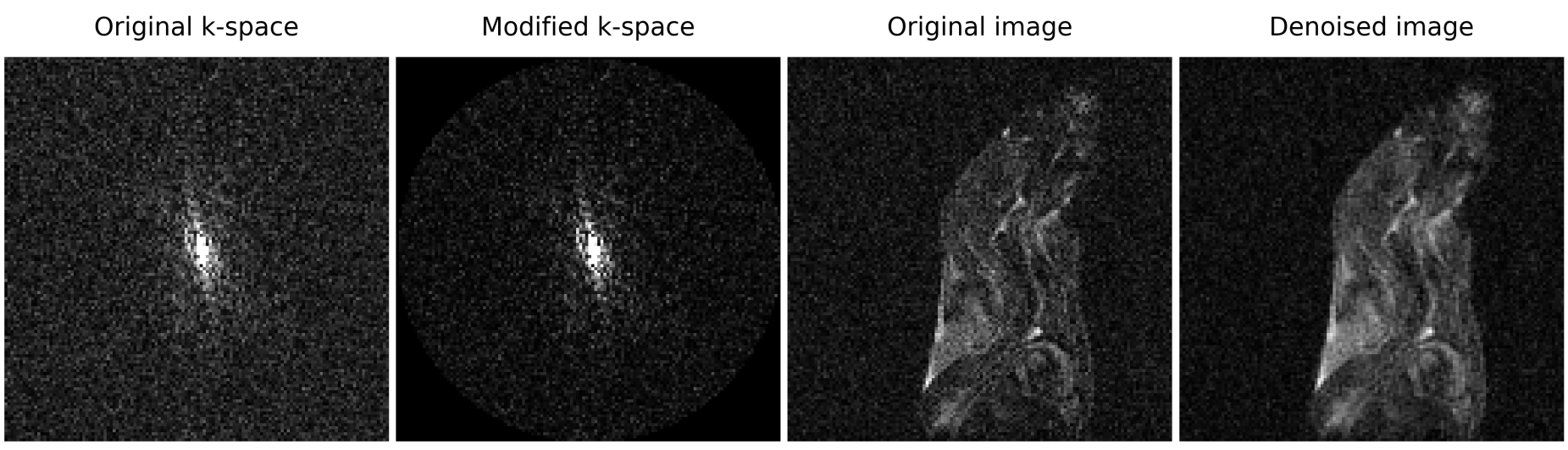
- weight_power value optimal for denoising without brightening.

In [8]:
```
wght_msk_kspace('mems_20190406_02.fid', 384, 119, 0.09, 1);
```



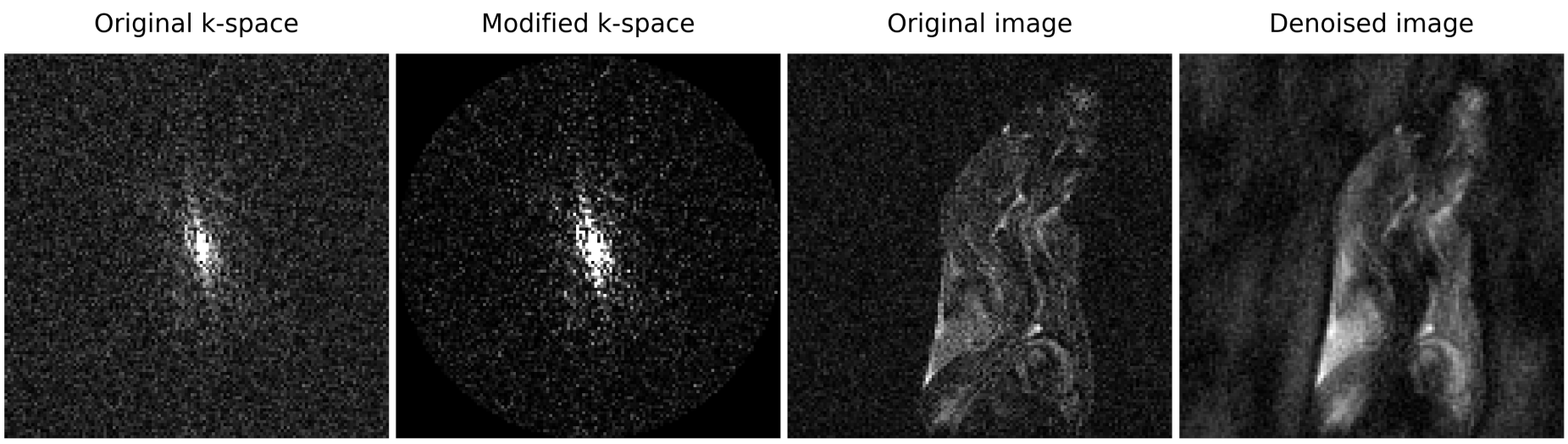
- It is possible to obtain a bit brighter good quality images using a bit higher weight_power value.

In [9]:
```
wght_msk_kspace('mems_20190406_02.fid', 384, 119, 0.14, 1);
```



- weight_power value too large for denoising - image artifacts.

In [10]:
```
wght_msk_kspace('mems_20190406_02.fid', 384, 119, 0.3, 1);
```



5 Literature reference (for the sample data)

Beata Wereszczyńska, Alcohol-fixed specimens for high-contrast post-mortem MRI, Forensic Imaging, Volume 25, 2021, 200449, ISSN 2666-2256, https://doi.org/10.1016/j.fri.2021.200449. (https://www.sciencedirect.com/science/article/pii/S2666225621000208)

6 About the author

Beata Wereszczyńska, PhD

e-mail: b.e.wereszczynska@gmail.com

GitHub: https://github.com/BeataWereszczynska

ORCID: https://orcid.org/0000-0003-3013-212X

ResearchGate: https://www.researchgate.net/profile/Beata-Wereszczynska

LinkedIn: https://www.linkedin.com/in/beata-wereszczynska/

7 License