دانشگاه صنعتی شریف
دانشکده مهندسی برق

عنوان:

گزارش پروژه درس

اعضای گروه

برنا قاضی زاده - ۴۰۱۱۰۲۲۸۶

نام درس

ریاضی مهندسی

نیم‌سال اول ۱۴۰۲-۱۴۰۳

نام استاد درس

دکتر کربلایی آقاجان

۱ MRI و $K-Space$

۱-۱ سوال ها

۱. تبدیل فوریه دو بعدی عملیاتی ریاضی است که تصویری را از دامنه‌ی فضای $x-y$ به دامنه‌ی فرکانسی می برد. به عبارت دیگر این تبدیل فرکانس های موجود در یک تصویر را تحلیل میکند. اگر بخواهیم با ریاضیات بررسی کنیم، تبدیل فوریه دو بعدی یک تصویر $f(x,y)$ را به $F(x,y)$ تبدیل میکند. در معادله‌ی زیر u و v مختصات دامنه فرکانس هستند.

$$F(u,v) = \iint f(x,y)e^{-i۲\pi(ux+vy)}\,dx\,dy$$

در اینجا، انتگرال بر روی کل بازه است.در نتیجه عملا $F(x,y)$ نمایانگر محتوای کل فرکانس فضای تصویر است. در $F(x,y)$ بزرگی آن نشان دهنده‌ی قدرت فرکانس های مختلف و فاز نشان دهنده‌ی شیفت فاز هر موئلفه‌ی فرکانسی است. تبدیل فوریه دو بعدی یک ابزار قدرتمند است که به ما این امکان را میدهد تا اطلاعات مفیدی درباره تصاویر از طریق تحلیل آنها در دامنه فرکانس بدست آوریم. این تبدیل در تحلیل و پردازش تصاویر پزشکی و فیلترینگ پرکاربرد است.
پایه های تبدیل فوریه دو بعدی ، توابع نمایی مختلط اند. تبدیل فوریه دو بعدی تصویر را به مجموعه ی از این توابع مختلط نمایی تجزیه میکند و محتوای فرکانس تصویر را نشان میدهد. هر تابع پایه یک فرکانس و یک جهت خاص را گرفته و همه با یکدیگر یک روش برای تجزیه و دستکاری تصاویر در دامنه فرکانس فراهم می‌کنند. هر تابع پایه یک فرکانس و یک جهت خاص را گرفته و همه با یکدیگر یک روش برای تجزیه و دستکاری تصاویر در دامنه فرکانس فراهم می‌کنند. این توابع پایه فرکانس‌های مختلف و جهت‌های مختلف را به خوبی در تصویر نشان می‌دهند. با همین روش می‌توان تصاویر را بر اساس محتوای فرکانسی آن‌ها تجزیه و تحلیل کرد. این رویکرد در بسیاری از حوزه‌های مهم از جمله پردازش تصویر پزشکی و فیلترینگ تصویر استفاده می‌شود. تحلیل تصاویر در دامنه فرکانس می‌تواند اطلاعات مفیدی را ارائه دهد که به طور معمول در تحلیل تصاویر در دامنه زمان به سادگی دسترسی نداریم. به عنوان مثال، می‌توان با استفاده از تحلیل فرکانسی یک تصویر پزشکی، اطلاعاتی در مورد ساختارهای بافتی داخلی یک بافت را بدست آورد که از دید زمان امکان‌پذیر نبوده است.

۲. هر نقطه در $k-space$ بیانگر نقطه ای در عکس ما است که با تبدیل فوریه و تبدیل فوریه

معکوس میتوان بین این دو فضا جابه‌جا شد.
هر نقطه در $k - space$ فرکانس در هر جهت را به ما میدهد.

۳. در عکس های فوق:
a متعلق به نقطه‌ی بالایی بر روی محور $K - y$ است.(چون فقط در راستای y حرکت میکند و از طرفی چون فرکانسش در این راستا از d بیشتر است پس نقطه بالایی برای عکس a شد.
با توجه به توضیحات فوق، عکس d نیز متعلق به نقطه‌ی پایینی بر روی محور $K - y$ است.
برای عکس b چون فقط در راستای افقی تغییرات دارد پس نقطه ای که روی محور K_x است متعلق به آن است.
نقطه‌ی آخر نیز میرسد به عکس d.

۴. حذف فرکانس‌های بالا در فضای k در زمان جمع‌آوری اطلاعات تصویربرداری MRI ، منجر به از دست رفتن وضوح فضایی در تصویر نهایی می‌شود. اجزای فرکانس بالا در فضای k، جزئیات دقیق و انتقالات تیز در تصویر را کد می‌کنند. حذف آن‌ها ممکن است منجر به محو شدن و کاهش وضوح تصویر شود. انجام تنظیمات مناسب بین زمان جمع‌آوری اطلاعات و کیفیت تصویر در این موارد حیاتی است.
حذف فرکانس‌های پایین در فضای k در زمان جمع‌آوری اطلاعات تصویربرداری MRI می‌تواند منجر به از دست رفتن کلیت تفاوت‌ها و اطلاعات مرتبط با ساختارهای کم‌تغییر شود. اجزای فرکانس پایین در فضای k به روشنایی کلی و تفاوت‌های تصویر کمک می‌کنند. حذف آن‌ها ممکن است منجر به از دست رفتن جزئیات آناتومیک مهم شود و تشخیص بین بافت‌های مختلف را دشوارتر کند. در تنظیم اجزاء فرکانسی در زمان جمع‌آوری تصاویر، تأثیر آن را بر کیفیت تصویر با دقت موردنظر قرار داده مهم است.
حفظ همزمان اجزاء فرکانس بالا و پایین در فضای k به طور کلی به حفظ تصویر اصلی بیشتر کمک می‌کند. اجزاء فرکانس بالا جزئیات دقیق را ضبط می‌کنند، در حالی که اجزاء فرکانس پایین به کلیت تفاوت‌ها و ساختار کلی اطلاعات اضافه می‌کنند. حفظ تعادل بین این اجزاء در زمان جمع‌آوری اطلاعات تصویربرداری MRI برای حفظ صحت و وضوح تصویر اصلی حائز اهمیت است. تنظیمات باید با دقت بر اساس اهداف خاص تصویربرداری و تضادهای بین وضوح فضایی و تفاوت تصویر انجام شود.
به طور خلاصه:
اجزاء فرکانس بالا:
جزئیات دقیق: این اجزاء اطلاعات مرتبط با جزئیات کم‌اندازه و تیز در تصویر را حاصل

می‌کنند.

اجزاء فرکانس پایین:

ساختار کلی و کنتراست: این اجزاء به تعیین کلیت تفاوت‌ها و ساختار کلی تصویر کمک میکنند.

حفظ کنتراست اطلاعات: حفظ این اجزاء موجب حفظ کنتراست کلی و اطلاعات مرتبط با ساختارهای کم‌تغییر در تصویر می‌شود.

Eng_math

January 30, 2024

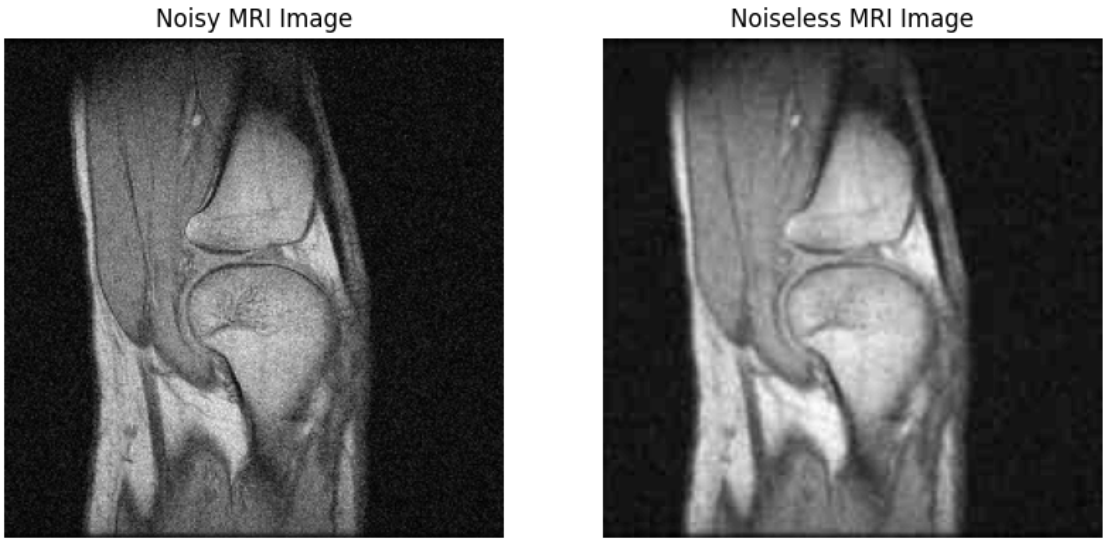
Part 3 @ Part 4:

[168]:
```
import scipy.io
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import imageio.v2 as iio
from scipy.signal import medfilt
from skimage import io, img_as_float
from skimage.restoration import denoise_nl_means
from scipy.ndimage import convolve
import matplotlib.pyplot as plt
from skimage import io, color
import cv2
```

[281]:
```
def calculate_snr(img1, img2):
    img1 = np.array(img1, dtype=np.float64)
    img2 = np.array(img2, dtype=np.float64)
    mean_img1 = np.mean((img1 ** 2))
    mse = np.mean((img1 - img2) ** 2)
    snr = 10 * np.log10(mean_img1 / mse)
    return snr

def calculate_psnr(img1, img2):
    mse = np.mean((img1 - img2) ** 2)
    max_pixel_value = 255.0
    psnr = 20 * np.log10(max_pixel_value / np.sqrt(mse))
    return psnr
```

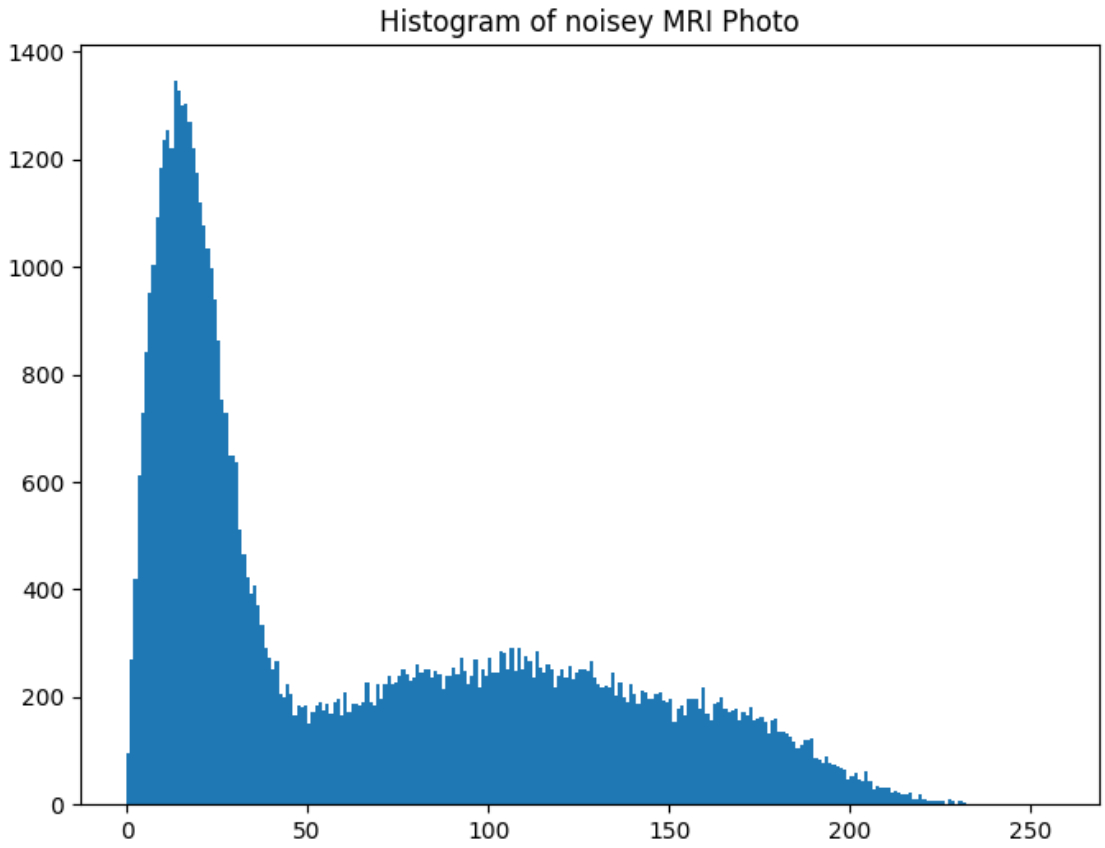
[283]:
```
mat_data = scipy.io.loadmat('rawkneedata.mat')
k_space_data = mat_data['dat']
reconstructed_image = np.fft.ifft2(k_space_data)
reconstructed_image_magnitude = np.abs(reconstructed_image)
shifted_reconstructed_image = np.fft.fftshift(reconstructed_image_magnitude)
noisy = np.array(shifted_reconstructed_image, dtype=np.float64)

img1_path = 'knee.png'
img2_path = 'kneeMRI.png'
```

```
img1 = np.array(iio.imread(img1_path), dtype=np.float64)
img2 = np.array(iio.imread(img2_path), dtype=np.float64)
# Plot the reconstructed MRI image
fig = plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(noisy, cmap='gray')
plt.axis('off')  # Turn off axis
plt.title('Noisy MRI Image')
plt.subplot(1, 2, 2)
plt.imshow(img2, cmap='gray')
plt.axis('off')  # Turn off axis
plt.title('Noiseless MRI Image')
plt.show()
print(f"The SNR value : {calculate_snr(img1, img2)} dB")
print(f"The PSNR value : {calculate_psnr(img1, img2)} dB")
```

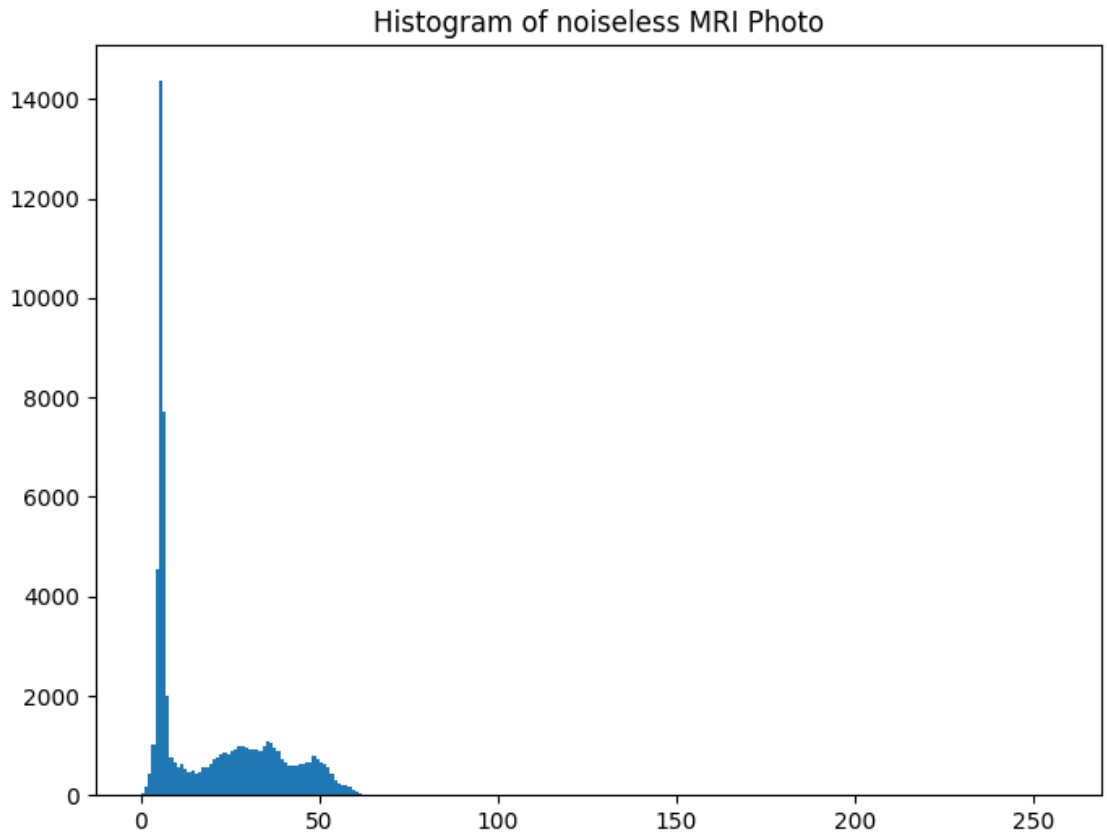



```
The SNR value : 2.916509323033147 dB
The PSNR value : 12.16490039737391 dB
```

[209]:
```
image = Image.open('knee.png')
image_array = np.array(image)
plt.figure(figsize=(8, 6))
plt.title('Histogram of noisey MRI Photo')
plt.hist(image_array.ravel(), 256, [0, 256])
plt.show()
```



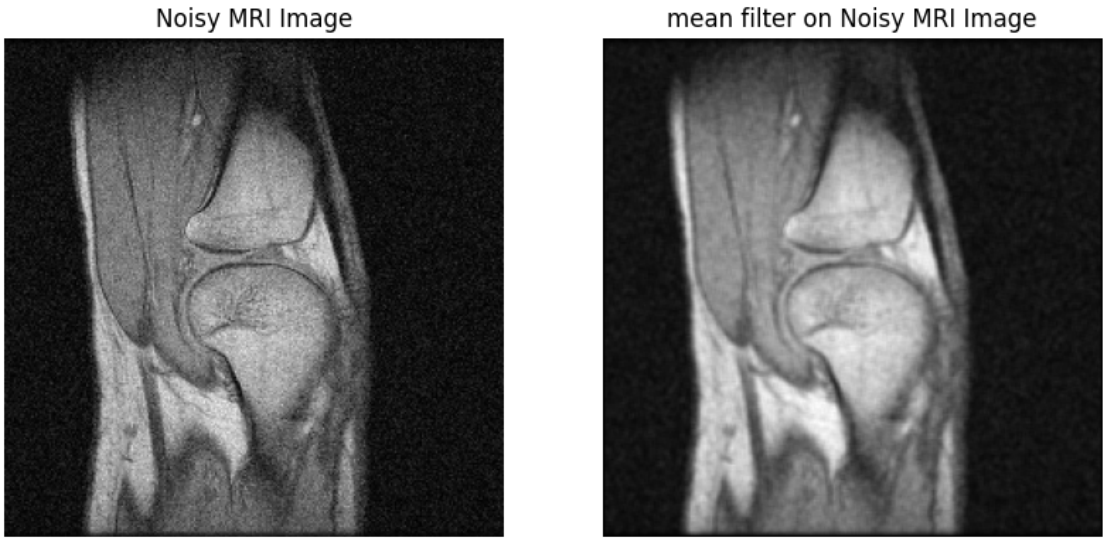
[55]:
```
image = Image.open('kneeMRI.png')
image_array = np.array(image)
plt.figure(figsize=(8, 6))
plt.title('Histogram of noiseless MRI Photo')
plt.hist(image_array.ravel(), 256, [0, 256])
plt.show()
```



[ ]:
```
##Mean Filter
```

[284]:
```
def apply_mean_filter(input_image):
    image_array = np.array(input_image)
    height, width = image_array.shape
    filtered_image_array = np.zeros((height, width), dtype=np.uint8)
    for i in range(1, height-1):
        for j in range(1, width-1):
            mean_value = np.mean(image_array[i-1:i+2, j-1:j+2])
            filtered_image_array[i, j] = mean_value.astype(np.uint8)

    filtered_image = Image.fromarray(filtered_image_array)
    return filtered_image

img1_path = 'mean.png'
img2_path = 'kneeMRI.png'
img1 = iio.imread(img1_path)
img2 = iio.imread(img2_path)
filtered_image = apply_mean_filter(np.array(np.array(iio.imread('knee.png'),␣
 ↪dtype=np.float64), dtype=np.float64))
```

```
fig = plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(shifted_reconstructed_image, cmap='gray')
plt.axis('off')  # Turn off axis
plt.title('Noisy MRI Image')
plt.subplot(1, 2, 2)
plt.imshow(filtered_image, cmap='gray')
plt.axis('off')  # Turn off axis
plt.title('mean filter on Noisy MRI Image')
plt.show()

print(f"The SNR value : {calculate_snr(filtered_image, img2)} dB")
print(f"The PSNR value : {calculate_psnr(filtered_image, img2)} dB")
```



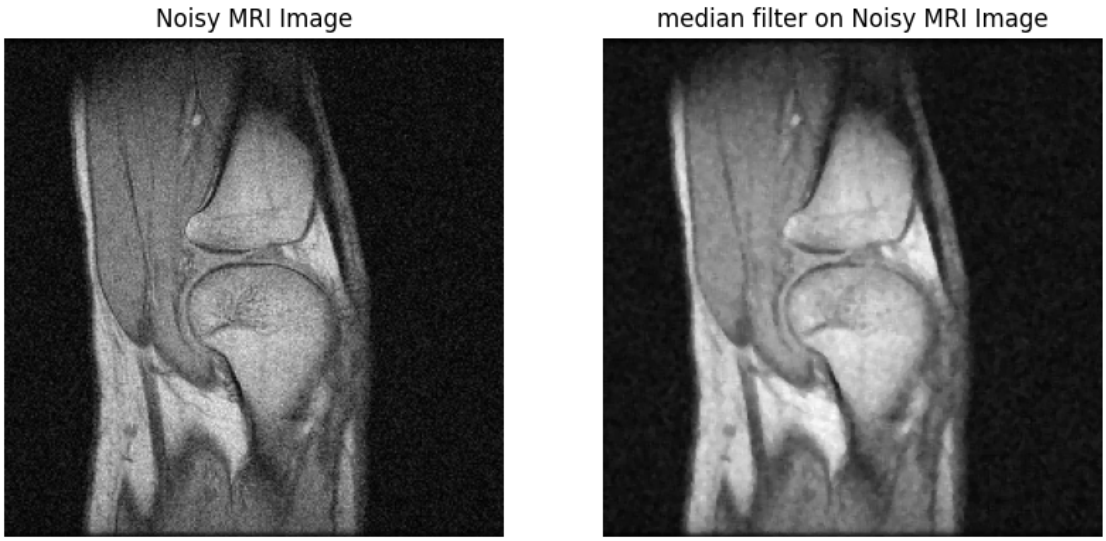

```
The SNR value : 3.029807915694389 dB
The PSNR value : 27.59967729093033 dB
```

[ ]:

[ ]:
```
##Median Filter
```

[285]:
```
def apply_median_filter(image, kernel_size):
    if len(np.array(image).shape) > 2:
        image = np.array(image.convert('L'))
    filtered_image = medfilt(image, kernel_size)
    return filtered_image
input_image = Image.open('knee.png')
filtered_image = apply_median_filter(input_image, kernel_size=3)
```

```
filtered_imagee = np.array(filtered_image, dtype=np.float64)
fig = plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(shifted_reconstructed_image, cmap='gray')
plt.axis('off')  # Turn off axis
plt.title('Noisy MRI Image')
plt.subplot(1, 2, 2)
plt.imshow(filtered_image, cmap='gray')
plt.axis('off')  # Turn off axis
plt.title('median filter on Noisy MRI Image')
plt.show()
print(f"The SNR value : {calculate_snr(filtered_imagee, img2)} dB")
print(f"The PSNR value : {calculate_psnr(filtered_image, img2)} dB")
```

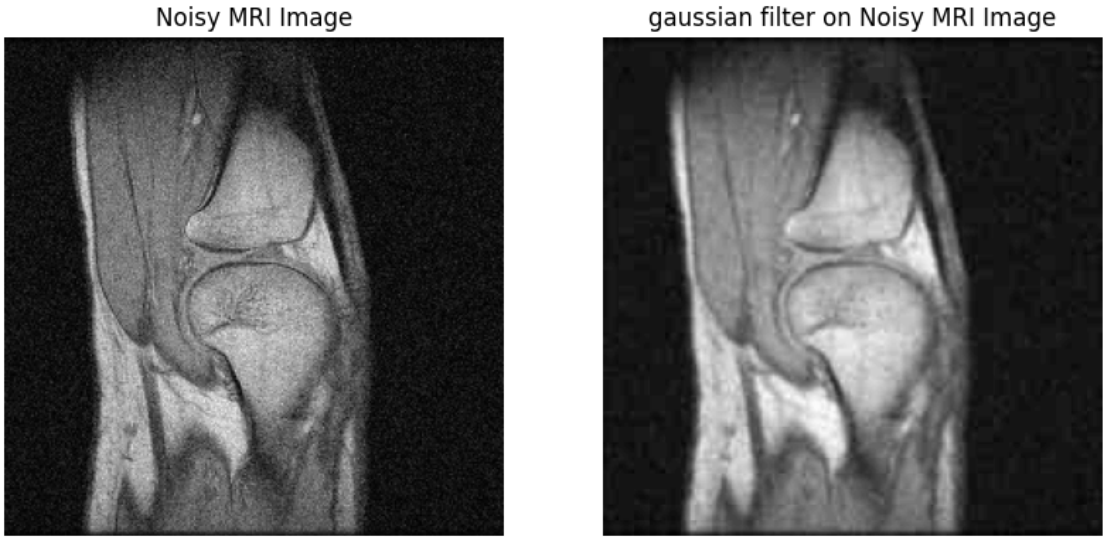



```
The SNR value : 3.0053442607225653 dB
The PSNR value : 27.83852310118251 dB
```

[ ]:
```
##Gaussian Filter
```

[290]:
```
def gaussian_kernel(size, sigma):
    kernel = np.fromfunction(
        lambda x, y: (1/(2*np.pi*sigma**2)) * np.exp(-((x - size//2)**2 + (y - size//2)**2)/(2*sigma**2)),
        (size, size)
    )
    return kernel / np.sum(kernel)

def apply_gaussian_filter(image, kernel_size, sigma):
    """Apply a Gaussian filter to the input image."""
```

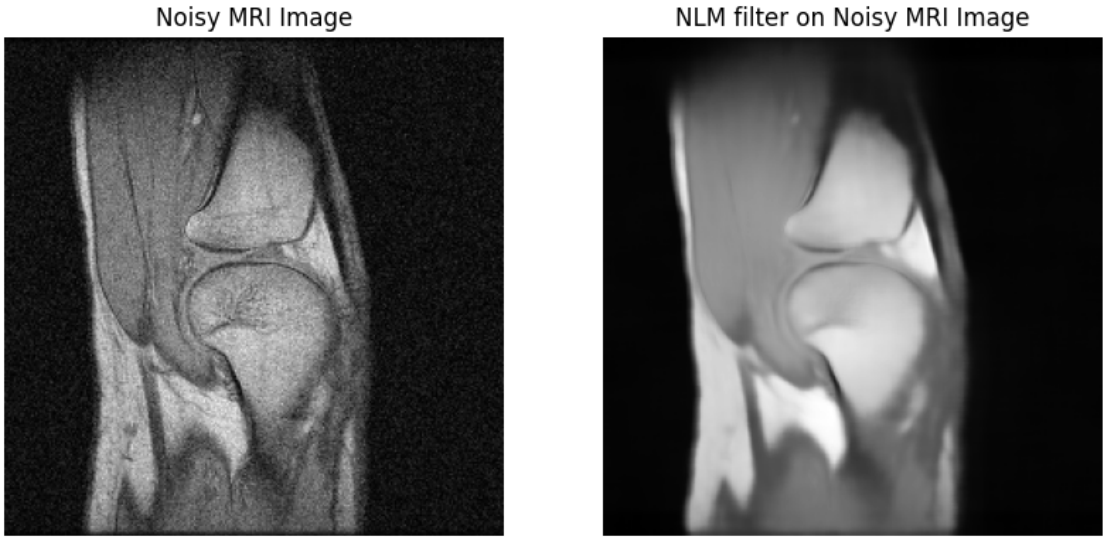
```
    if len(image.shape) > 2:
        image = color.rgb2gray(image)
    kernel = gaussian_kernel(kernel_size, sigma)
    filtered_image = convolve(image, kernel)

    return filtered_image

img1_path = 'gaussian.png'
img2_path = 'kneeMRI.png'
img1 = iio.imread(img1_path)
img2 = iio.imread(img2_path)
img3_path = 'knee.png'
img3 = iio.imread(img3_path)

filtered_image = apply_gaussian_filter(img3, kernel_size=5, sigma=1.5)

fig = plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(shifted_reconstructed_image, cmap='gray')
plt.axis('off')  # Turn off axis
plt.title('Noisy MRI Image')
plt.subplot(1, 2, 2)
plt.imshow(img2, cmap='gray')
plt.axis('off')  # Turn off axis
plt.title('gaussian filter on Noisy MRI Image')
plt.show()
print(f"The SNR value : {calculate_snr(filtered_image, img2)} dB")
print(f"The PSNR value : {calculate_psnr(filtered_image, img2)} dB")
```



```
The SNR value : 3.044914180804965 dB
The PSNR value : 27.31573654289403 dB
```

[ ]:
```
##NLM filter
```

[294]:
```
image = io.imread('knee.png')
img1_path = 'nlm.png'
img2_path = 'kneeMRI.png'
img1 = iio.imread(img1_path)
img2 = iio.imread(img2_path)
image = img_as_float(image)
denoised_image = denoise_nl_means(image, h=0.1)
denoised_image = np.array(denoised_image, dtype=np.float64)
fig = plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(shifted_reconstructed_image, cmap='gray')
plt.axis('off')  # Turn off axis
plt.title('Noisy MRI Image')
plt.subplot(1, 2, 2)
plt.imshow(denoised_image, cmap='gray')
plt.axis('off')  # Turn off axis
plt.title('NLM filter on Noisy MRI Image')
plt.show()
print(f"The SNR value : {calculate_snr(np.array(denoised_image, dtype=np.
 ↪float64), img2)} dB")
print(f"The PSNR value : {calculate_psnr(np.array(denoised_image, dtype=np.
 ↪float64), img2)} dB")
```




```
The SNR value : -37.41453294849331 dB
```

```
The PSNR value : 20.125437598303076 dB
```