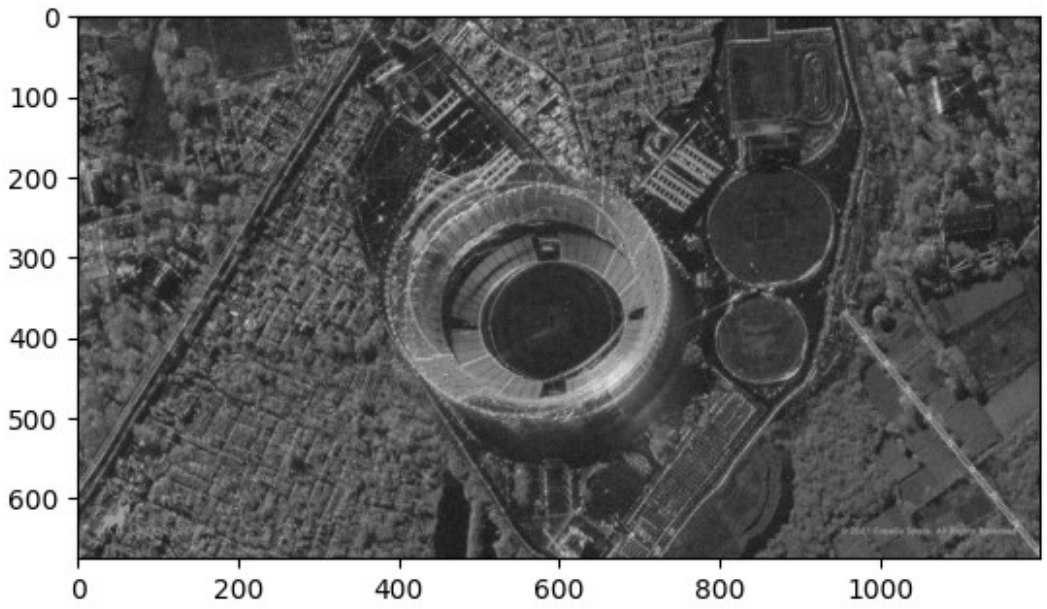
```
import numpy as np
import cv2
import matplotlib.pyplot as plt

image = cv2.imread('sar_1.jpg')
image_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
plt.imshow(image_gray, cmap="gray")
plt.show()
```



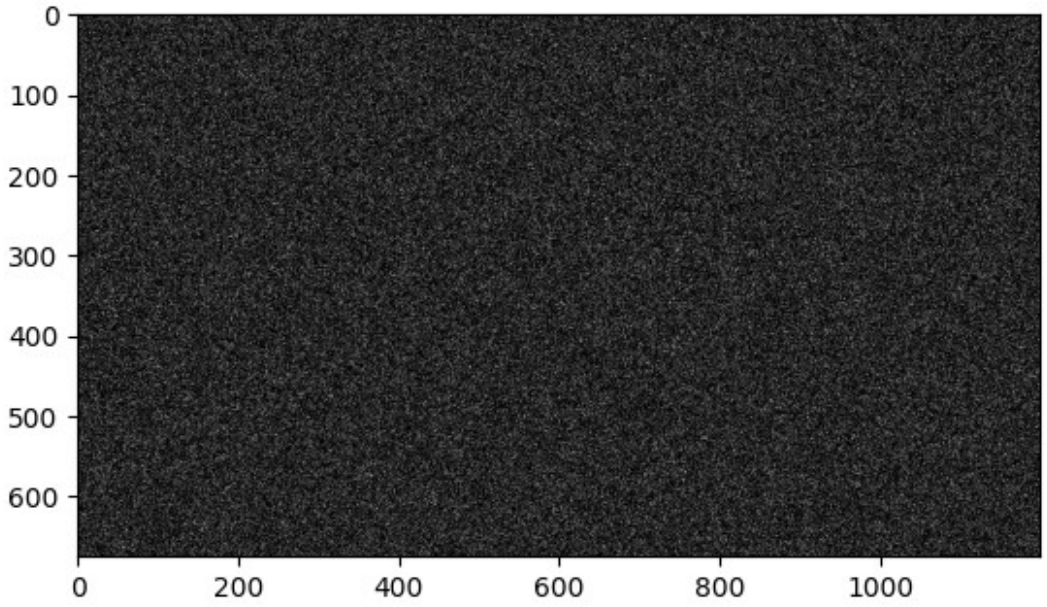

```
# ДЗ 2
# Зашумить изображение при помощи шума гаусса, постоянного шума.
# Протестировать медианный фильтр, фильтр гаусса, билатериальный
фильтр, фильтр нелокальных средних с различными параметрами.
# Выяснить, какой фильтр показал лучший результат фильтрации шума.
# Зашумить изображение 2 при помощи шума типа соль-перец.
# Исследовать, как влияет частота шума на качество работы
морфологических операций открытие/закрытие.
```

Зашумить изображение при помощи шума гаусса, постоянного шума.

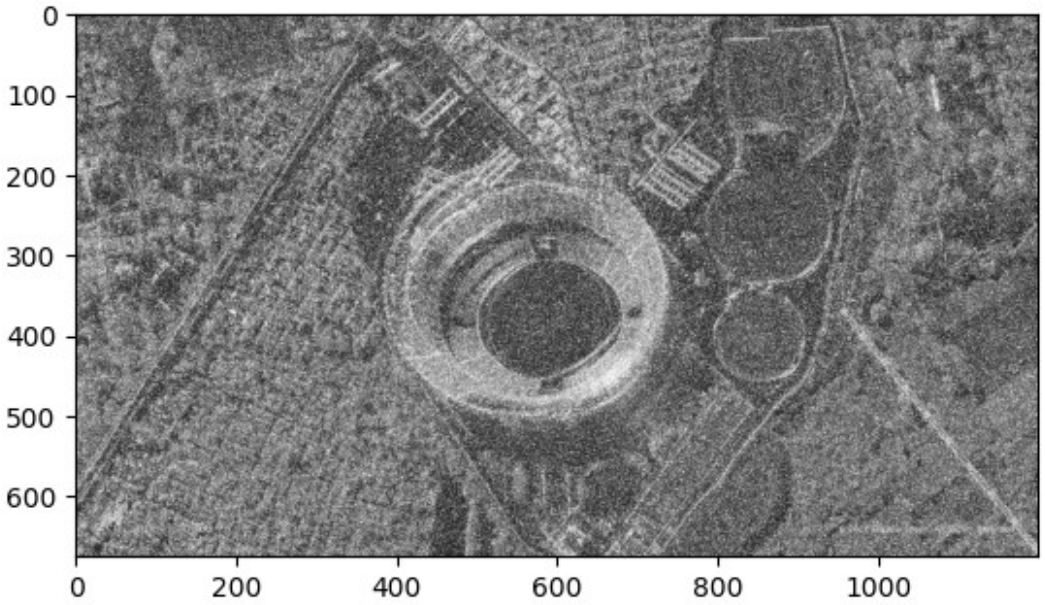
```
# Gaussian noise
mean = 0
stddev = 100
noise_gauss = np.zeros(image_gray.shape, np.uint8)
cv2.randn(noise_gauss, mean, stddev)
```

```
array([[ 23,   0,  52, ...,  58,   0,  52],
       [ 47,   0,   0, ...,   0,   0,   0],
       [  0,  25, 119, ...,  17,  56,  57],
       ...,
       [190,  15,   0, ..., 119,  51,   0],
       [  0,   0,   0, ...,   0,   0,   0],
       [ 45,  49,   0, ...,  29,   0,   0]], dtype=uint8)
```

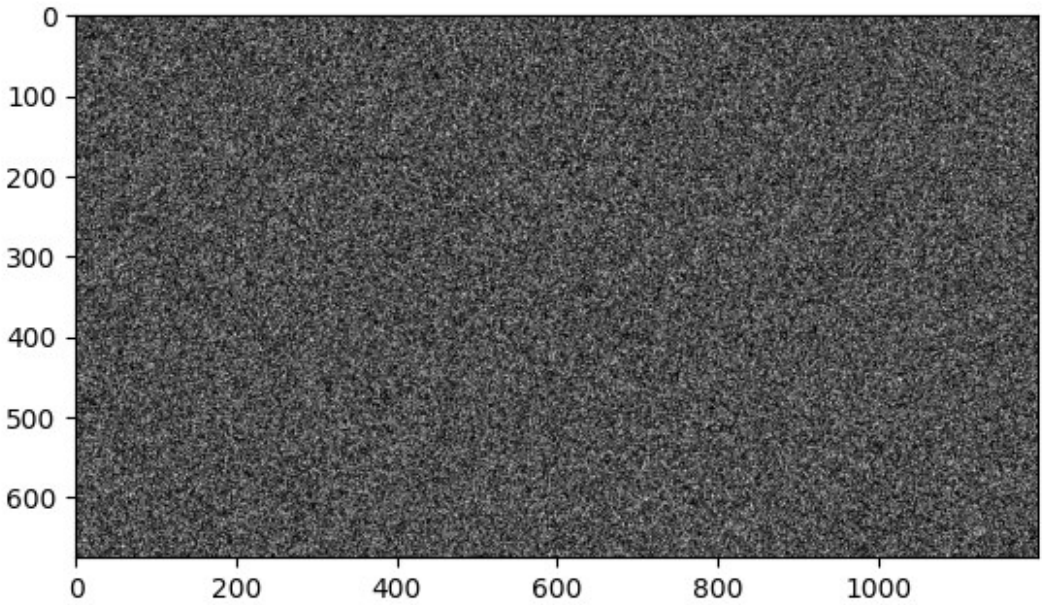
```
plt.imshow(noise_gauss, cmap="gray")
plt.show()
```




```
image_noise_gauss = cv2.add(image_gray,noise_gauss)  # наложение шума
на изображение
plt.imshow(image_noise_gauss, cmap="gray")
plt.show()
```

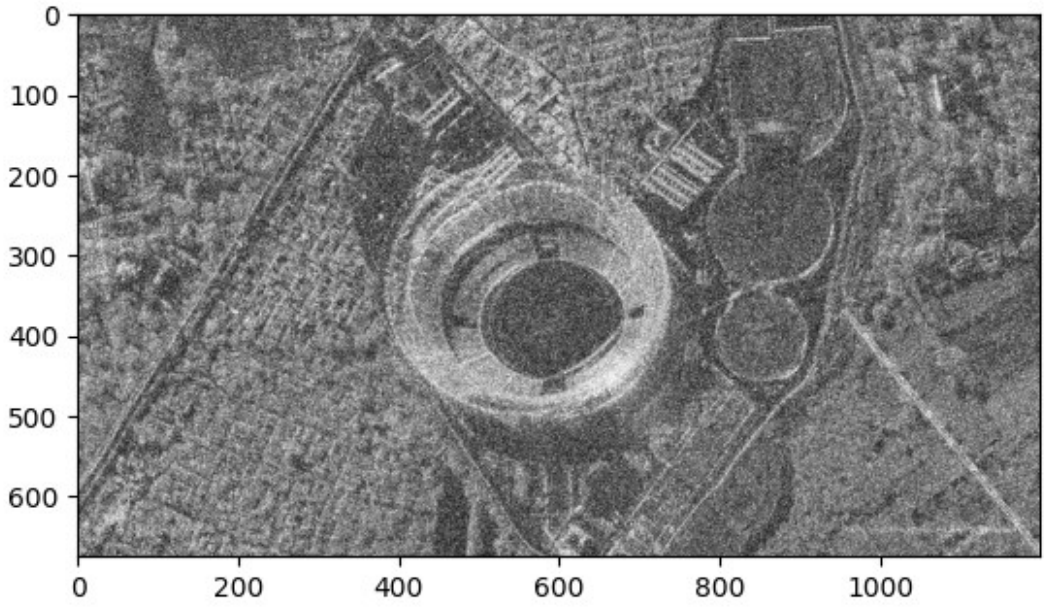



```
# Постоянный шум
uni_noise=np.zeros(image_gray.shape, np.uint8)
cv2.randn(uni_noise,0,255)
uni_noise=(uni_noise*0.5).astype(np.uint8)
plt.imshow(uni_noise, cmap="gray")
plt.show()
```




```
image_nois_uni = cv2.add(image_gray,uni_noise) # наложение шума на изображение
```

```
plt.imshow(image_nois_uni, cmap="gray")
plt.show()
```



Протестировать медианный фильтр, фильтр гаусса, билатериальный фильтр, фильтр нелокальных средних с различными параметрами.

```
# Фильтр Гаусса
from skimage.metrics import structural_similarity, mean_squared_error

# Медианный фильтр
image_gauss_median = cv2.medianBlur(image_noise_gauss, 3)

# Билатеральный фильтр
image_gauss_bilat = cv2.bilateralFilter(image_noise_gauss, 9, 75, 75)

# Фильтр нелокальных средних с различными параметрами
image_gauss_nlm = cv2.fastNlMeansDenoising(image_noise_gauss, h=20)
```

Выяснить, какой фильтр показал лучший результат фильтрации шума.

```
# Оценка результатов фильтрации
# Для каждого фильтра рассчитываем SSIM и MSE

mse_gauss = mean_squared_error(image_gray, image_noise_gauss)
(ssim_gauss, _) = structural_similarity(image_gray, image_noise_gauss,
full=True)
print("Фильтр Гаусса MSE:", mse_gauss, "SSIM:", ssim_gauss)

mse_median = mean_squared_error(image_gray, image_gauss_median)
(ssim_median, _) = structural_similarity(image_gray,
image_gauss_median, full=True)
print("Медианный фильтр MSE:", mse_median, "SSIM:", ssim_median)

mse_bilat = mean_squared_error(image_gray, image_gauss_bilat)
(ssim_bilat, _) = structural_similarity(image_gray, image_gauss_bilat,
full=True)
print("Билатериальный фильтр MSE:", mse_bilat, "SSIM:", ssim_bilat)

mse_nlm = mean_squared_error(image_gray, image_gauss_nlm)
(ssim_nlm, _) = structural_similarity(image_gray, image_gauss_nlm,
full=True)
print("Фильтр нелокальных средних MSE:", mse_nlm, "SSIM:", ssim_nlm)
```

```
Фильтр Гаусса MSE: 4219.287662962963 SSIM: 0.1875406166110974
Медианный фильтр MSE: 1028.5219283950617 SSIM: 0.4297814165846413
Билатериальный фильтр MSE: 1831.0438543209877 SSIM:
0.31494925654994005
Фильтр нелокальных средних MSE: 4214.846219753086 SSIM:
0.1879542095638936
```