Practical – 3
Digital Image Processing Lab

Name: Utsav Balar
Roll.no. T24cs003

Observe various type of noise effect on images:

Code:

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

def plt_show(img: np.ndarray, dst: np.ndarray, t1: str, t2: str) →
None:
    plt.figure(figsize=(10, 8))
    plt.subplot(221)
    plt.title(t1)
    plt.imshow(img, cmap="gray")

    plt.subplot(222)
    plt.title(t2)
    plt.imshow(dst, cmap="gray")

    plt.subplot(223)
    plt.title(t1 + " profile")
    plt.plot(img[0, :])

    plt.subplot(224)
    plt.title(t2 + " profile")
    plt.plot(dst[0, :])

    plt.tight_layout()
    plt.show()


def non_local_means_denoising(img, params: tuple) → None:
    # h - Filter scale
    # templateWindowSize - Size of the averaging window
    # searchWindowSize - Size of the search window
    dst = cv.fastNlMeansDenoising(
        img, None, h=params[0], templateWindowSize=params[1],
searchWindowSize=params[2]
    )
    plt_show(img, dst, "Original", "Denoised")
```

```
def bilateral_denoising(img: np.ndarray, params: tuple) → None:
    # d - Diameter of each pixel neighborhood
    # sigmaColor - Filter sigma in color space
    # sigmaSpace - Filter sigma in coordinate space

    # Convert to 32f if it's not already
    img_32f = img.astype(np.float32)

    # Applying Bilateral Filter for denoising
    dst = cv.bilateralFilter(img_32f, d=params[0],
sigmaColor=params[1], sigmaSpace=params[2])

    plt_show(img, dst, "Original", "Bilateral Denoised")

def salt_and_pepperify(img: np.ndarray, params: tuple) →
np.ndarray:
    row, col = img.shape

    salt = np.random.rand(row, col) < params[0]
    pepper = np.random.rand(row, col) < params[1]

    noisy_image = np.copy(img)
    noisy_image[salt] = 1
    noisy_image[pepper] = 0

    return noisy_image


def poisson_noisify(img: np.ndarray) → np.ndarray:
    noise_mask = np.random.poisson(img)
    return img + noise_mask

def speckle_noisify(img: np.ndarray) → np.ndarray:
    noisy_img = img + 0.15 * img.std() *
np.random.randn(*img.shape)
    return noisy_img


img1 = cv.imread('1.png', cv.IMREAD_GRAYSCALE)
non_local_means_denoising(img1, (30, 7, 21))

img2 = cv.imread('3.png', cv.IMREAD_GRAYSCALE)
non_local_means_denoising(img2, (30, 7, 21))

# Gaussian Noise
```

```
img3 = cv.imread('2.png', cv.IMREAD_GRAYSCALE)
noisy_img = cv.GaussianBlur(img3, (45, 45), 0)
plt_show(img3, noisy_img, 'Original', 'Gaussian Noise')
# denoise using sharpening techniques

# Impulse Noise (Salt & Pepper)
img4 = cv.imread('2.png', cv.IMREAD_GRAYSCALE)
noisy_img = salt_and_pepperify(img4, (0.1, 0.01))
plt_show(img4, noisy_img, 'Original', 'Salt & Pepper
Noise')bilateral_denoising(noisy_img, (9, 75, 75))

# Poisson Noise
img5 = cv.imread("2.png", cv.IMREAD_GRAYSCALE)
noisy_img = poisson_noisify(img5)
plt_show(img5, noisy_img, "Original", "Poisson Noise")
bilateral_denoising(noisy_img, (9, 75, 75))

# Speckle Noiseimg6 = cv.imread("2.png", cv.IMREAD_GRAYSCALE)
noisy_img = speckle_noisify(img6)
plt_show(img6, noisy_img, "Original", "Speckle Noise")
bilateral_denoising(noisy_img, (9, 75, 75))
```

Results:

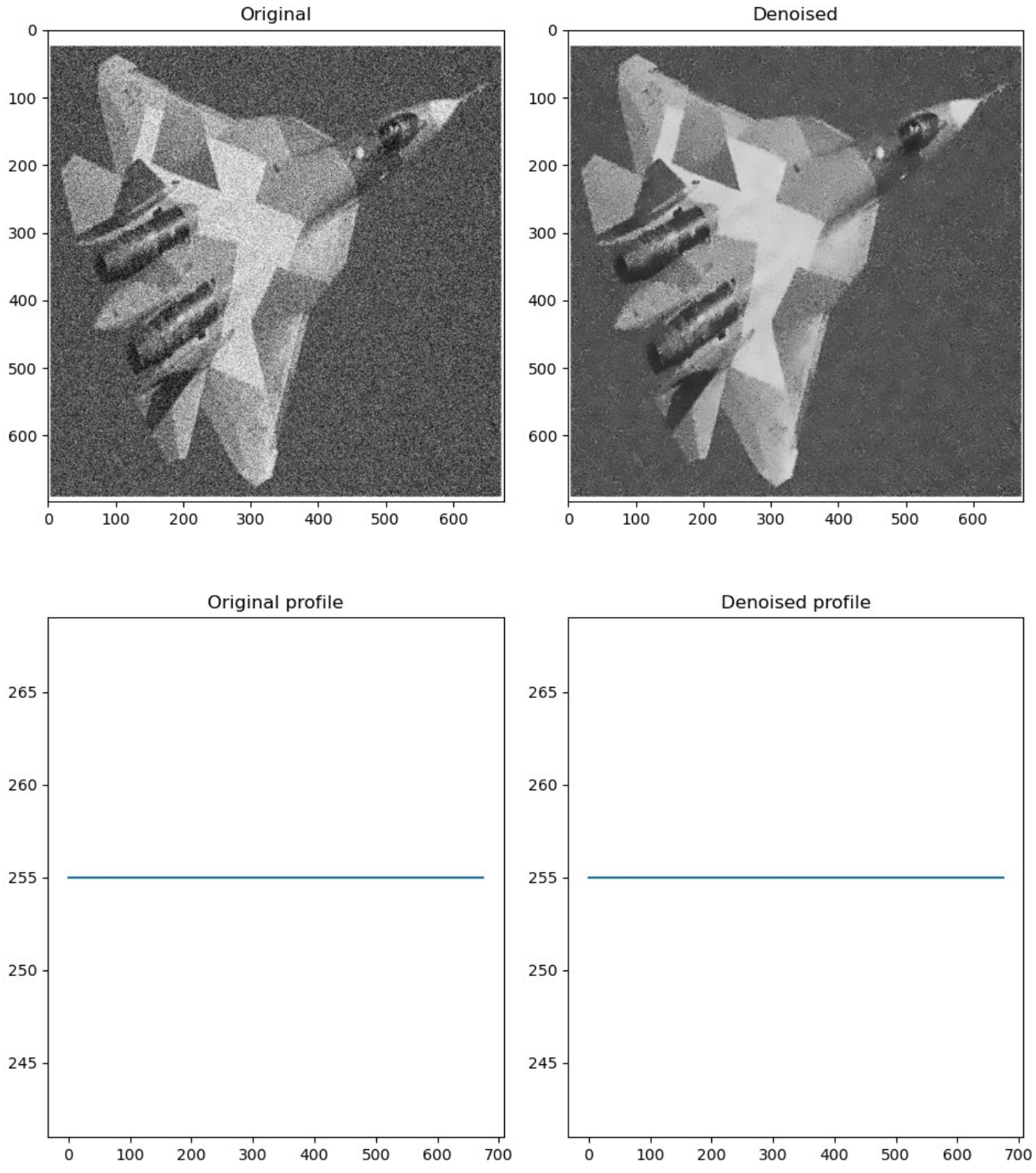

Fig1: Non Local means denoising on airplane image

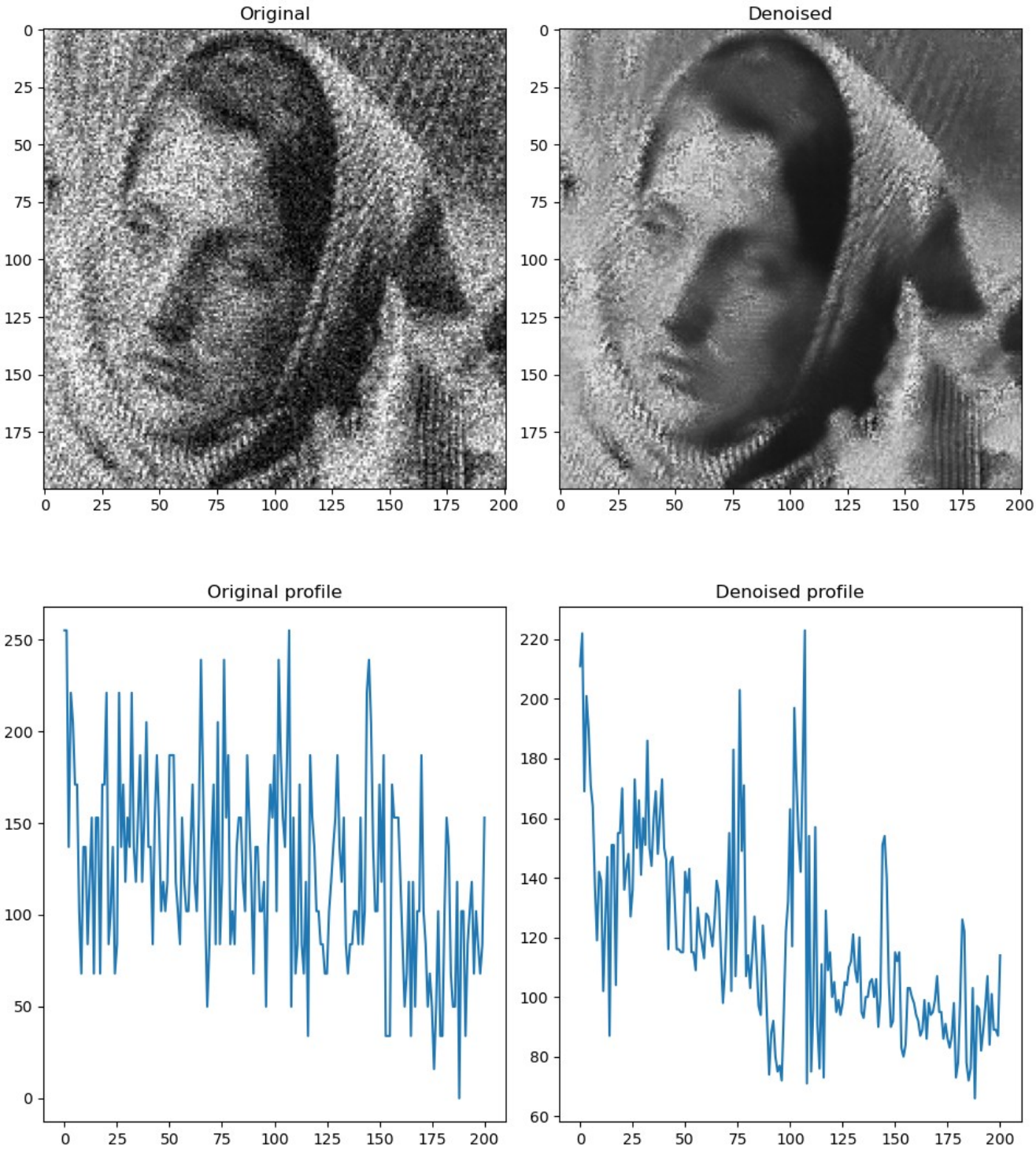

Fig2: Non Local means denoising on leya's image

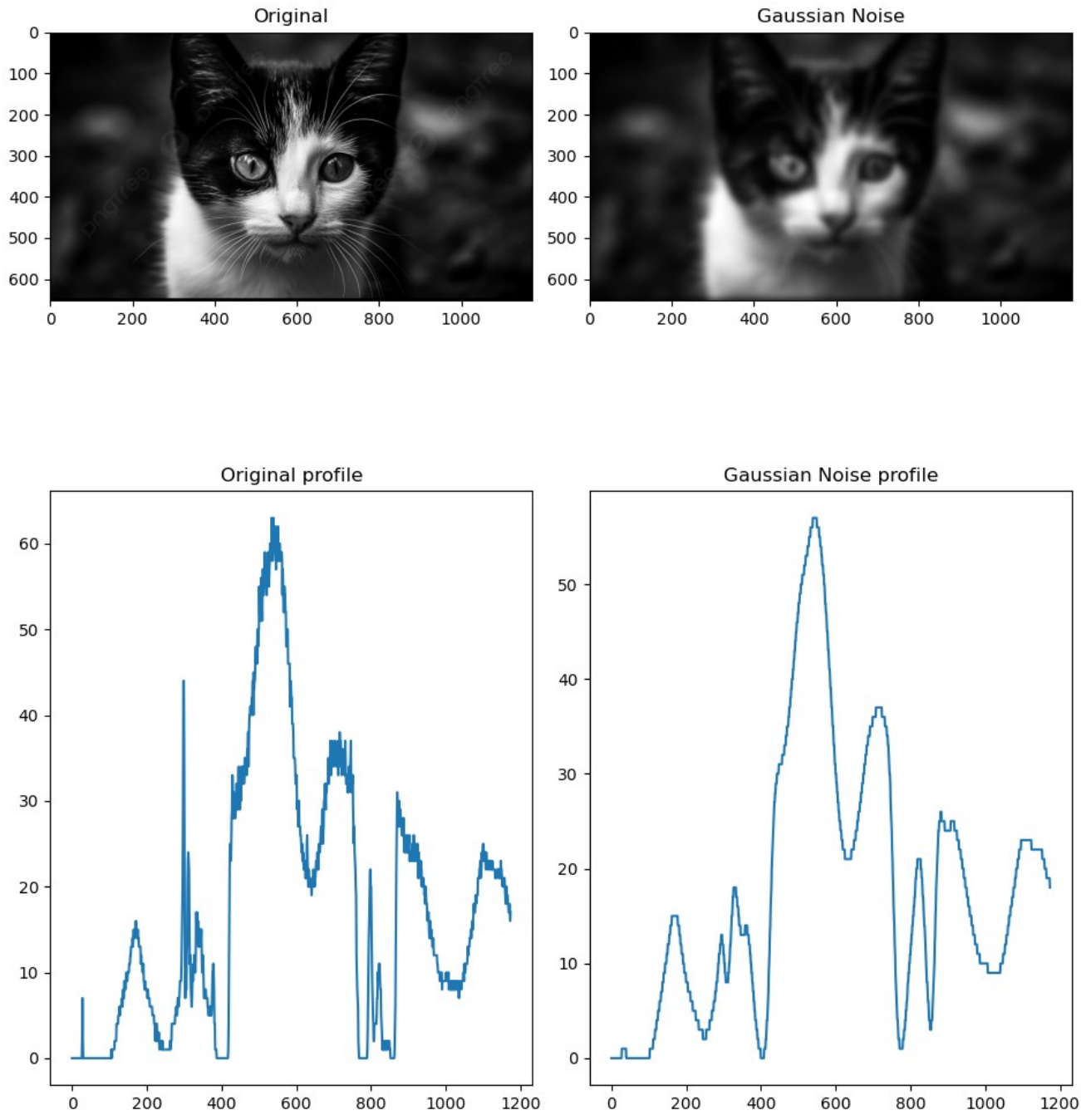

Fig3: Adding Gaussian noise

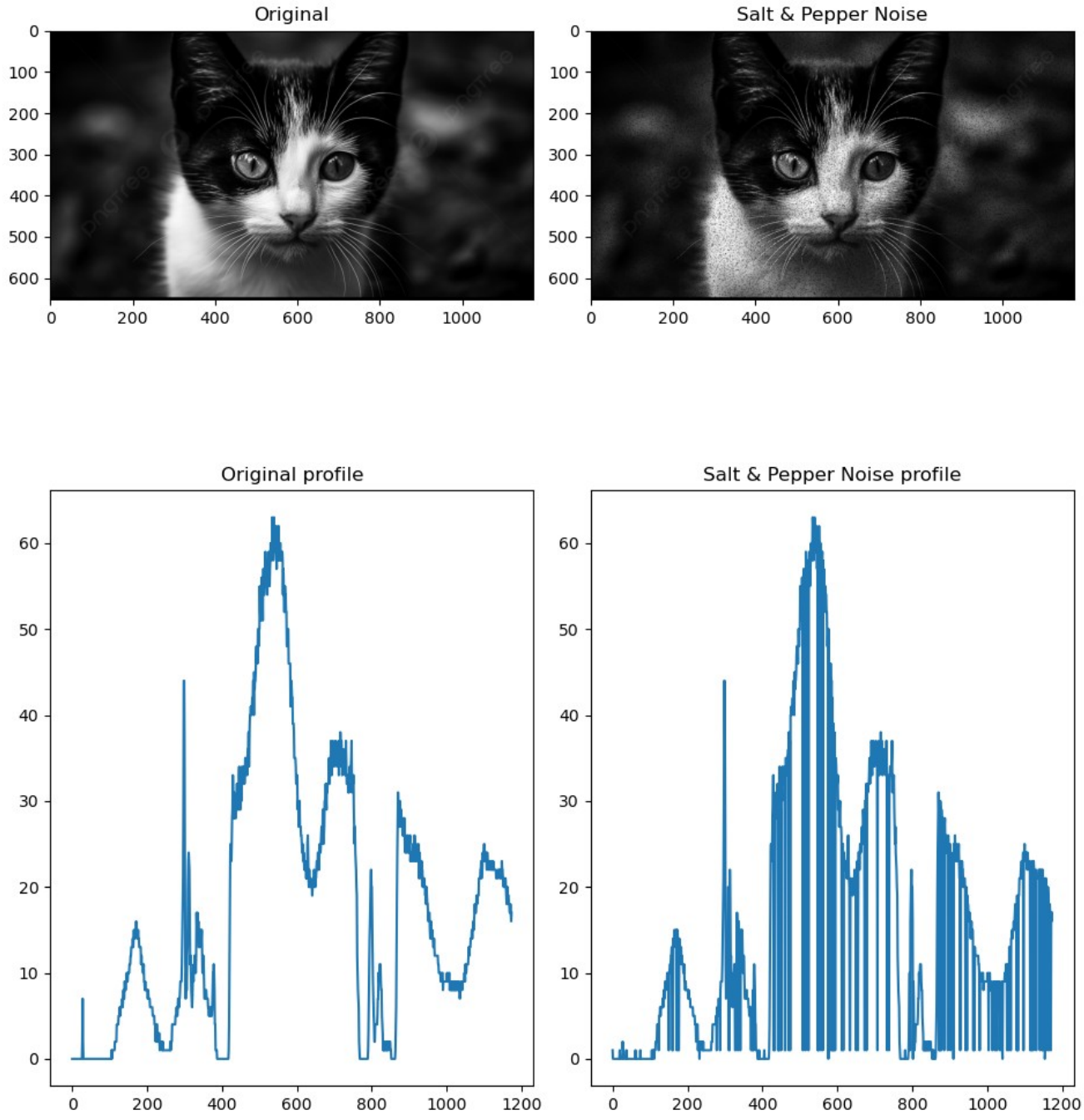

Fig4: Adding Salt and Pepper noise

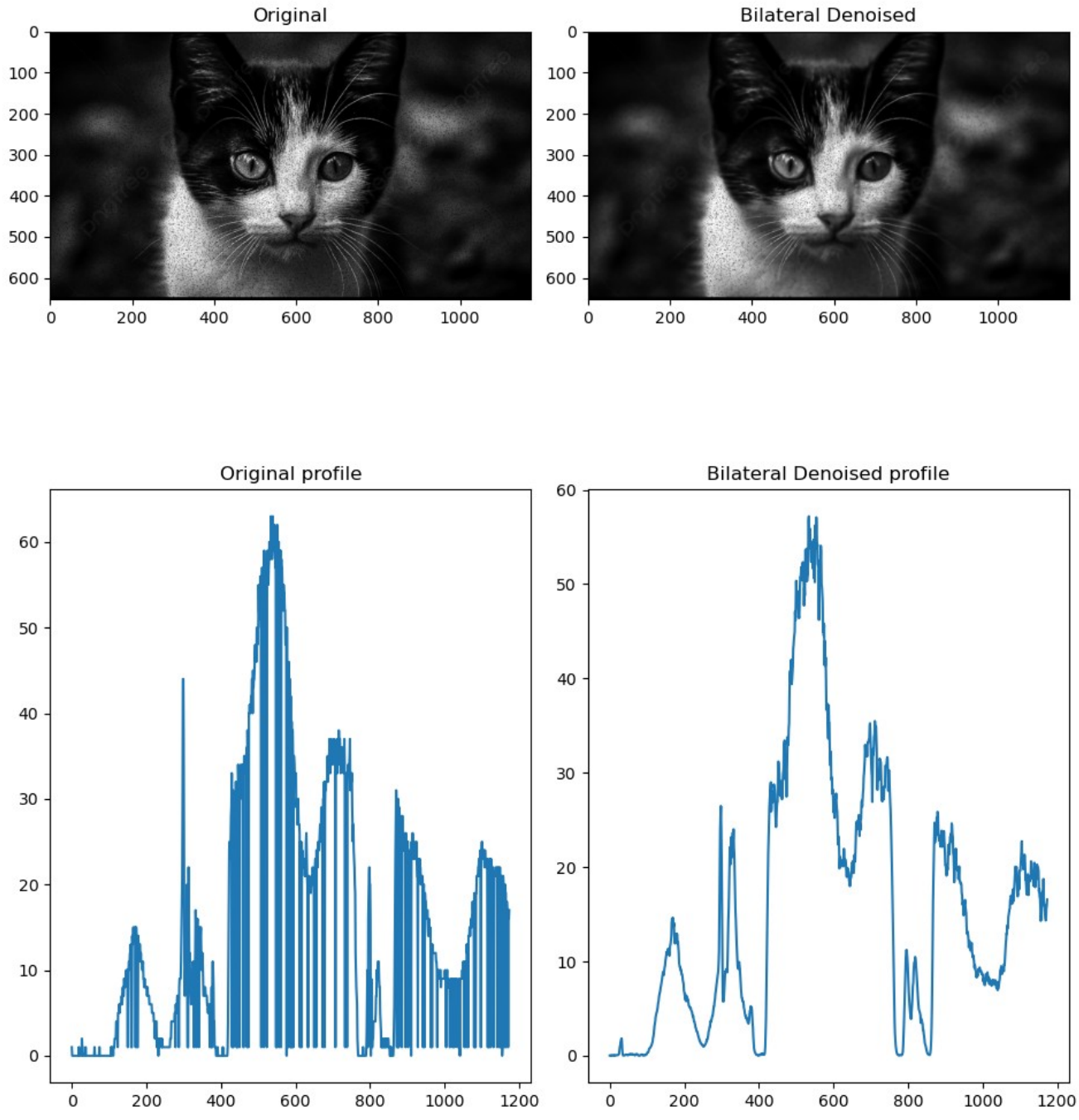

Fig5: Removing Salt and Pepper noise

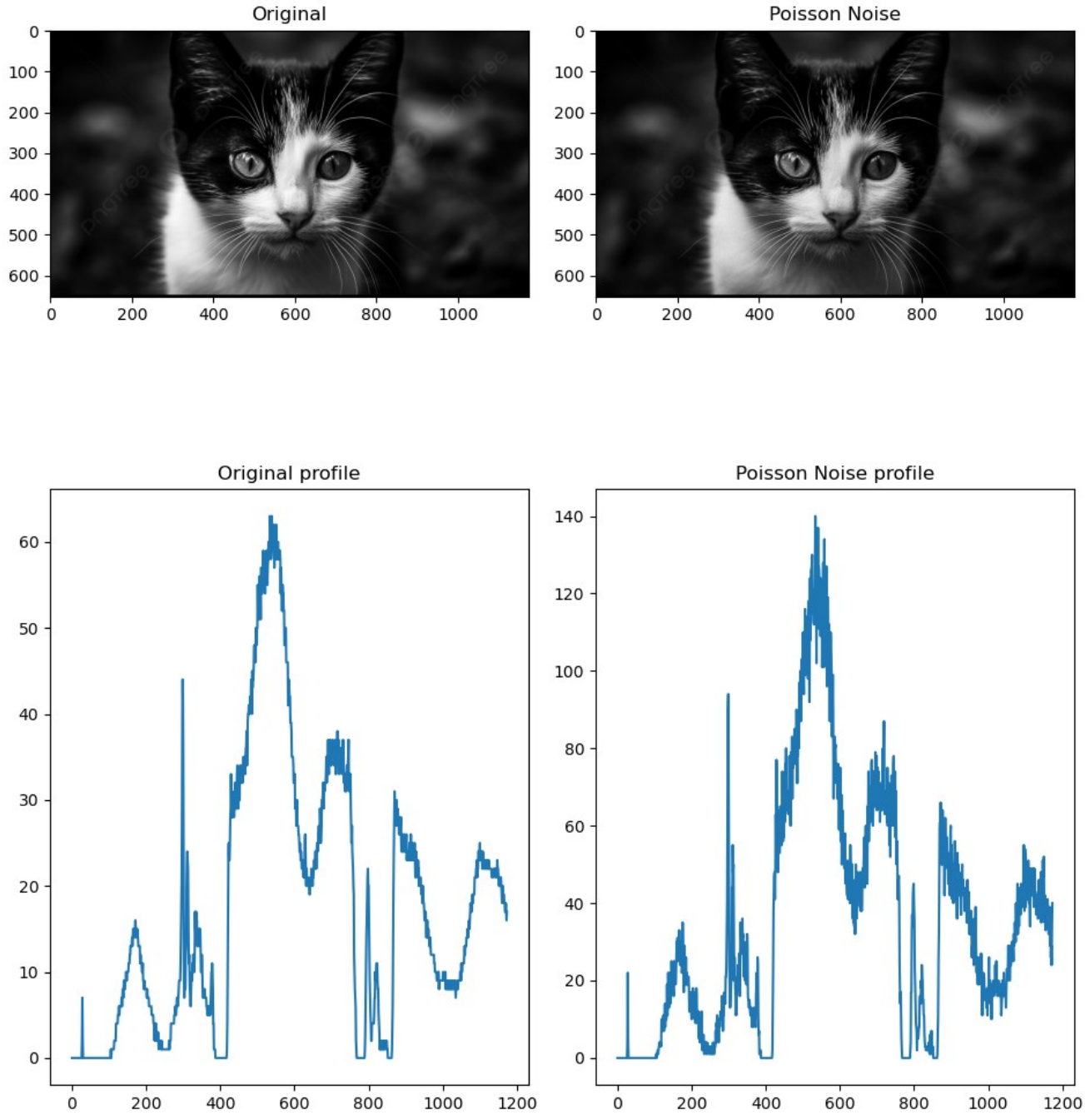

Fig6: Adding Poisson noise

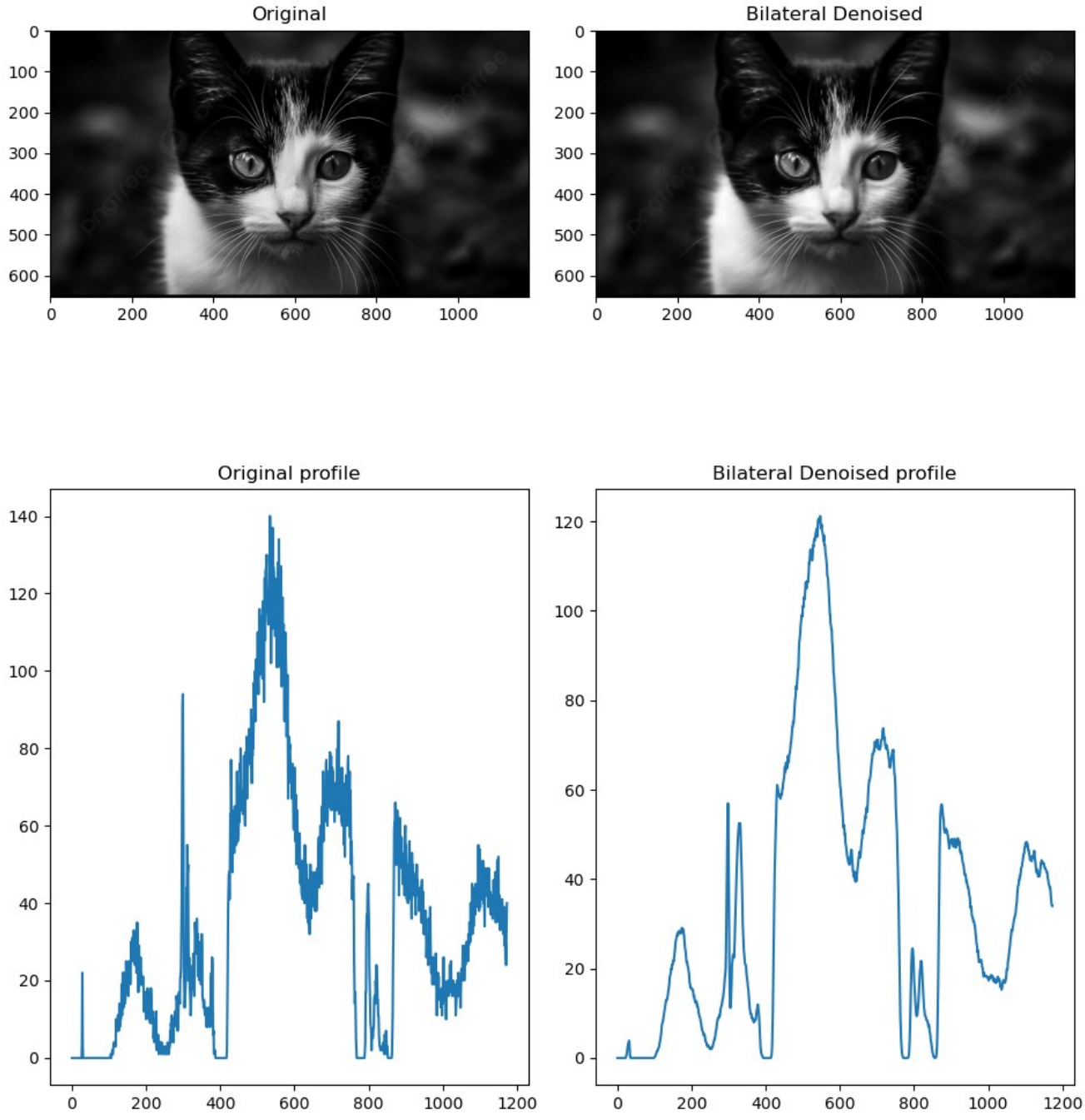

Fig7: Removing poisson noise

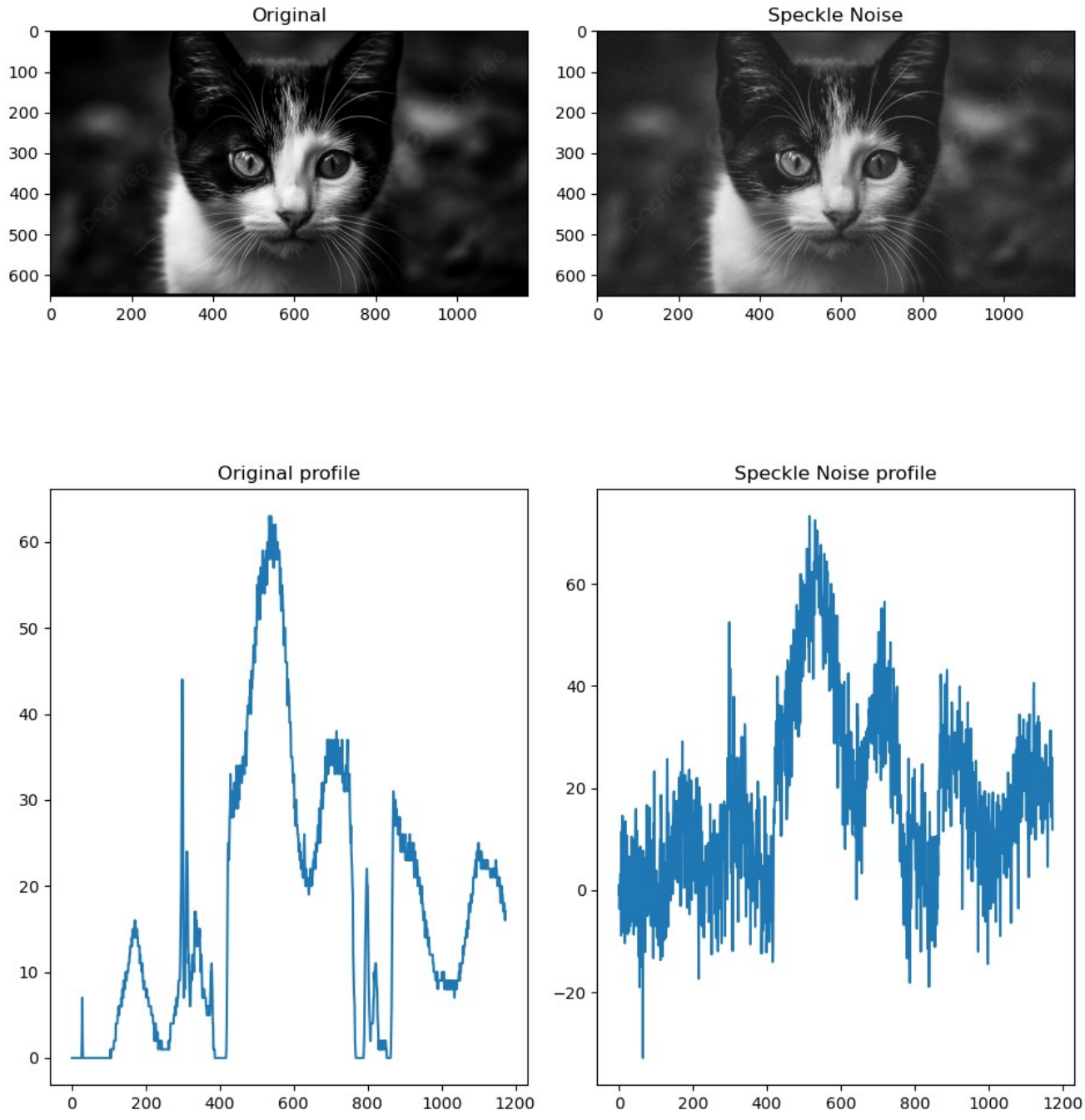

Fig8: Adding Speckle noise

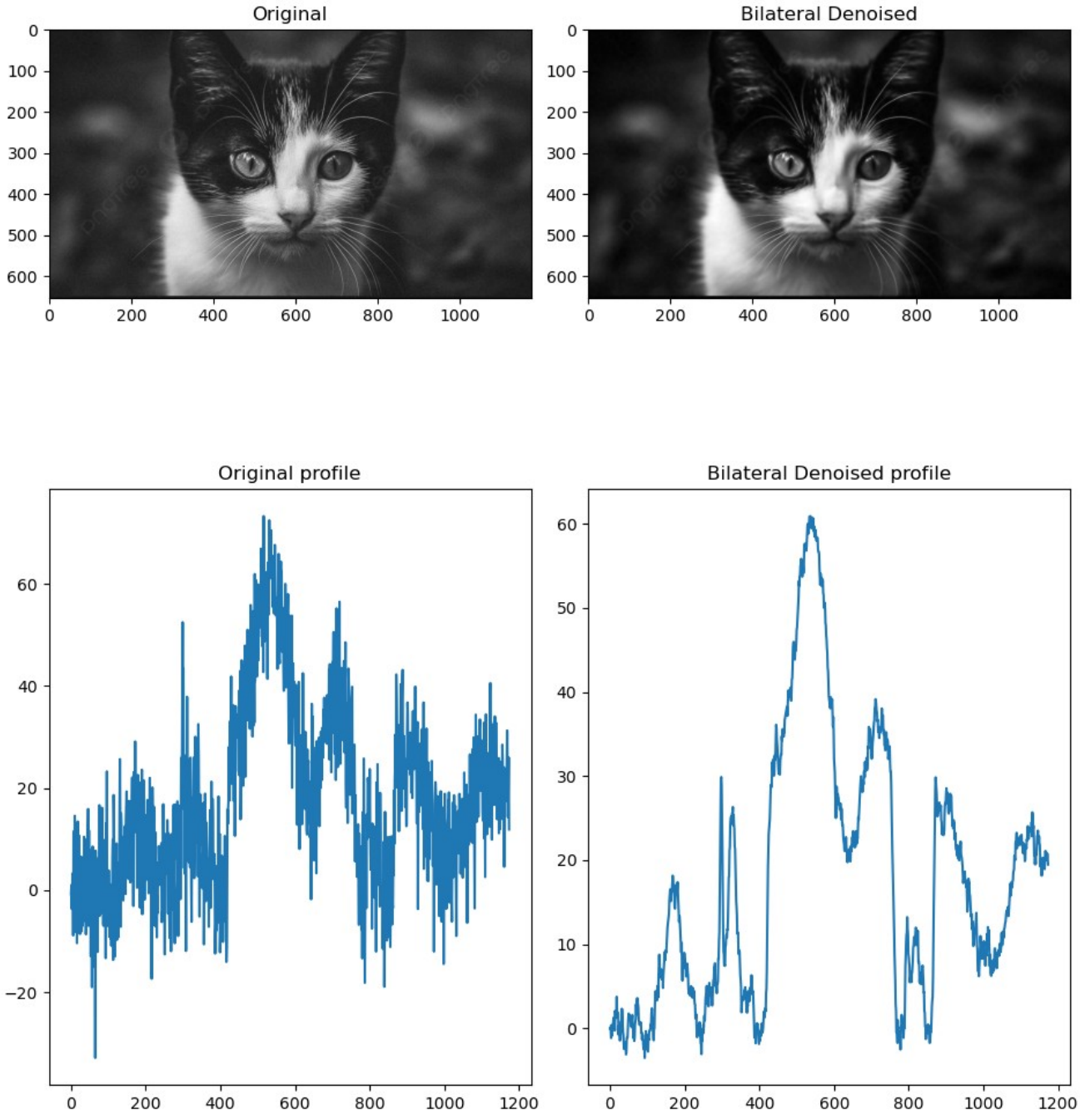

Fig9: Removing Speckle noise