1.Please use a self-designed Lowpass Gaussian Filter Kernels tocancel the shaded pattern noise of the image, 'checkerboard1024-

shaded.tif'. Please describe the your lowpass Gaussian filter kernels,

shading noise pattern as Fig. 3.42 (b), final corrected image without

shading noise pattern as Fig. 3.42 (c) and print out the source code? (40)

1. Lowpass Gaussian Filter Kernel

Gaussian Kernel Generation:

The program uses a 255x255 Gaussian kernel with a standard deviation (sigma) of 64. A Gaussian kernel is a smoothing filter designed to reduce high-frequency noise in an image while preserving lower-frequency structures or features.

Specifically, the Gaussian kernel is calculated using the Gaussian function, where each value is computed based on its distance from the center. The values closer to the center are higher, while those farther from the center decrease. When applied to an image, the kernel emphasizes the central region and blurs the surrounding areas.

Function of the Gaussian Kernel:

The main purpose of this kernel is to remove high-frequency details from the image and highlight lower-frequency background or broad changes. It performs smoothing, making the shading pattern more uniform.

Through the cv2.filter2D function, this filter is applied to the image, generating the "Shaded Pattern," which is the image after removing high-frequency noise.

2. Shading Noise Pattern

Shaded Pattern:

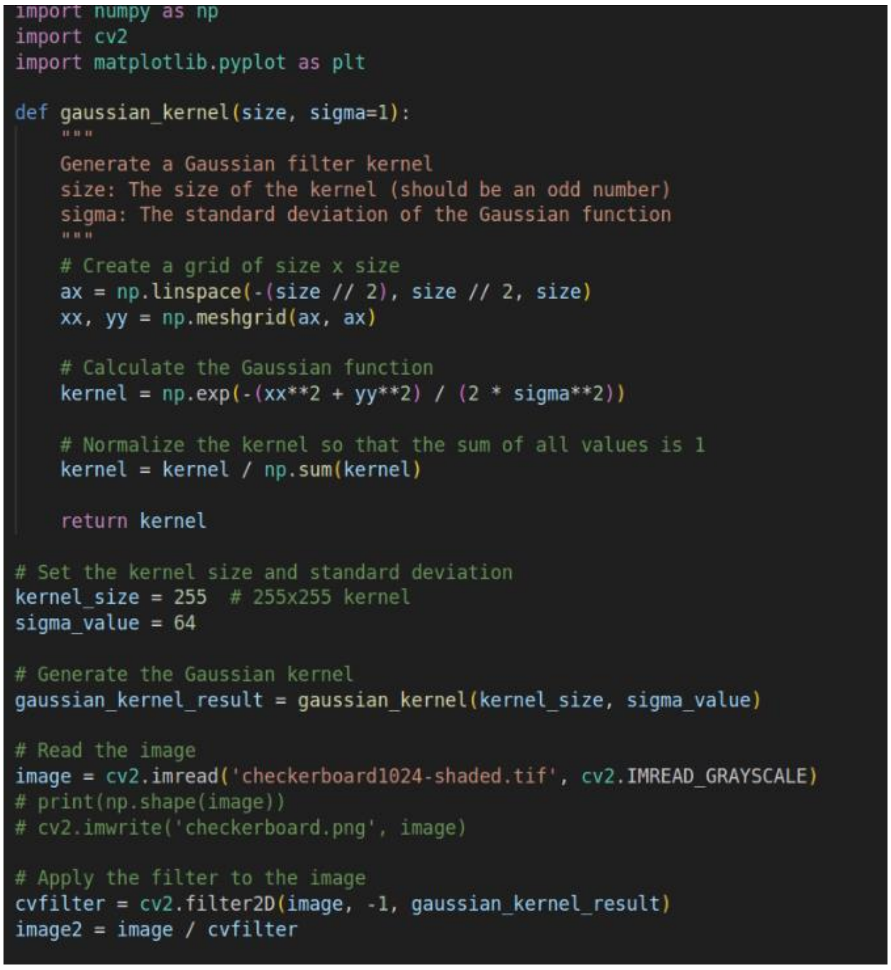
The application of the Gaussian kernel removes most high-frequency noise from the image, thus extracting a low-frequency shading or uneven lighting pattern from the image. This is referred to as the "Shaded Pattern," representing the background shadow in the original image caused by lighting variations or other light-related factors.

Image Correction:

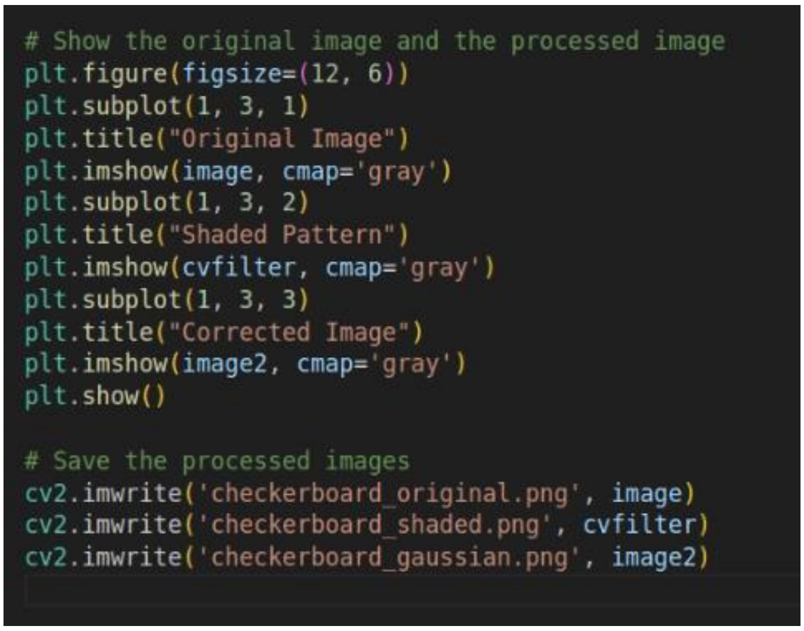
In the second step, the program divides the original image by this "Shaded Pattern" to generate a "Corrected Image." The purpose of this operation is to remove the shadow effect from the image, making it more uniform and clearer, thus reducing distortion caused by uneven lighting.

Summary:

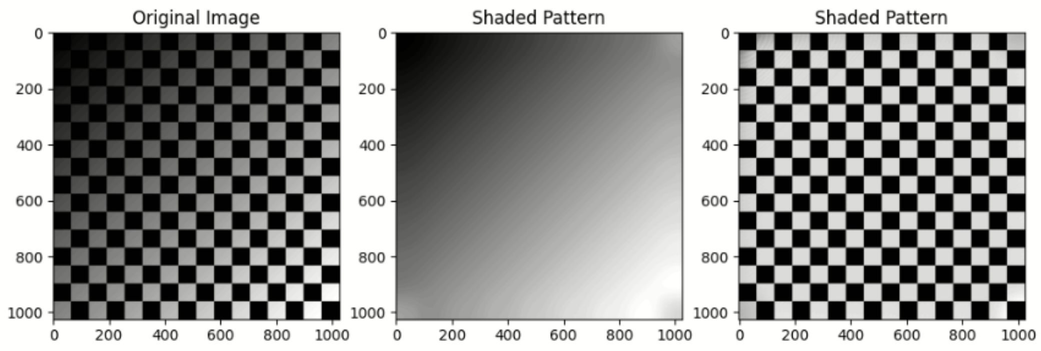
The 255x255 Gaussian filter used in the program effectively smooths the image, extracts the shading pattern, and removes uneven shadows from the image, achieving image correction. This is a typical application of a lowpass filter, mainly aimed at eliminating high-frequency noise and uneven lighting.



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

def gaussian_kernel(size, sigma=1):
    """
    Generate a Gaussian filter kernel
    size: The size of the kernel (should be an odd number)
    sigma: The standard deviation of the Gaussian function
    """
    # Create a grid of size x size
    ax = np.linspace(-(size // 2), size // 2, size)
    xx, yy = np.meshgrid(ax, ax)

    # Calculate the Gaussian function
    kernel = np.exp(-(xx**2 + yy**2) / (2 * sigma**2))

    # Normalize the kernel so that the sum of all values is 1
    kernel = kernel / np.sum(kernel)

    return kernel

# Set the kernel size and standard deviation
kernel_size = 255  # 255x255 kernel
sigma_value = 64

# Generate the Gaussian kernel
gaussian_kernel_result = gaussian_kernel(kernel_size, sigma_value)

# Read the image
image = cv2.imread('checkerboard1024-shaded.tif', cv2.IMREAD_GRAYSCALE)
# print(np.shape(image))
# cv2.imwrite('checkerboard.png', image)

# Apply the filter to the image
cvfilter = cv2.filter2D(image, -1, gaussian_kernel_result)
image2 = image / cvfilter
```



```
# Show the original image and the processed image
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.title("Original Image")
plt.imshow(image, cmap='gray')
plt.subplot(1, 3, 2)
plt.title("Shaded Pattern")
plt.imshow(cvfilter, cmap='gray')
plt.subplot(1, 3, 3)
plt.title("Corrected Image")
plt.imshow(image2, cmap='gray')
plt.show()

# Save the processed images
cv2.imwrite('checkerboard_original.png', image)
cv2.imwrite('checkerboard_shaded.png', cvfilter)
cv2.imwrite('checkerboard_gaussian.png', image2)
```



2. Repeat (1) steps in the image 'N1.bmp' (40)

1. Lowpass Gaussian Filter Kernel

Gaussian Kernel Generation:

The code uses a 201x201 Gaussian kernel with a standard deviation (sigma) set to 45. A Gaussian kernel is a smoothing filter primarily used to reduce high-frequency noise in an image while preserving low-frequency structures or features.

The kernel values are calculated using a Gaussian function. Values near the center are larger, and values further from the center are smaller. When applied to an image, this Gaussian kernel emphasizes the central area and blurs the surroundings, achieving a smoothing effect.

Function of the Gaussian Kernel:

The main function of this Gaussian kernel is to remove high-frequency details in the image, highlighting the low-frequency background or broader variations, which makes the shading pattern more uniform.

Using the cv2.filter2D function, this filter kernel is applied to the original image to produce the so-called "Shaded Pattern," which represents the image with reduced high-frequency noise.

2. Shading Noise Pattern

Shaded Pattern:

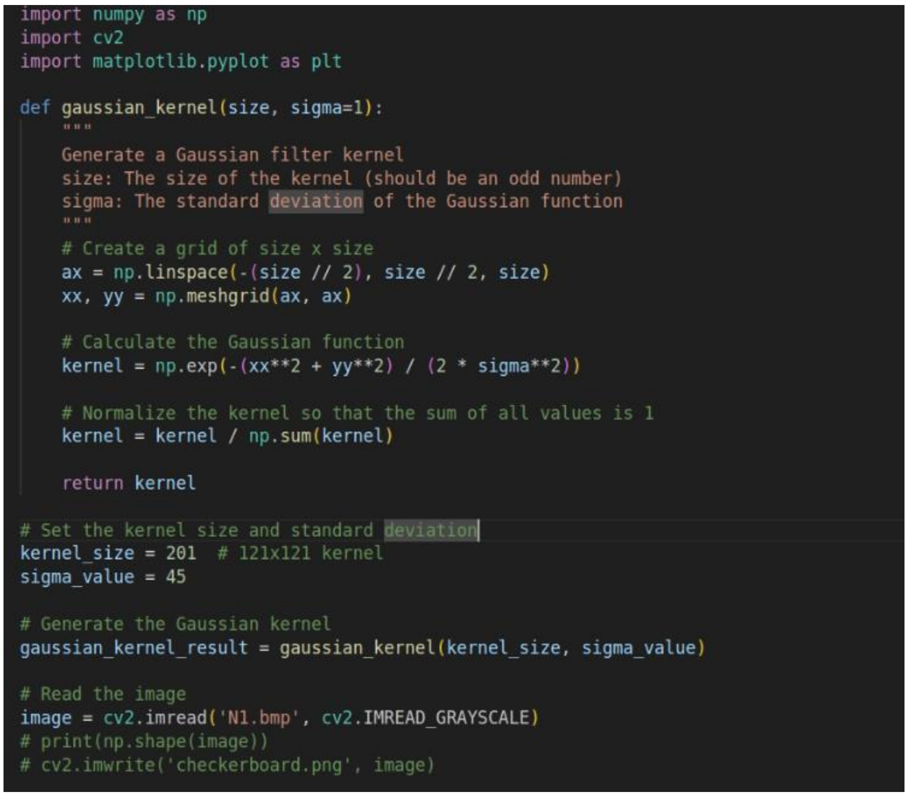
After applying the Gaussian kernel, most of the high-frequency noise in the image is removed, extracting the low-frequency shading or uneven lighting pattern, referred to as the "Shaded Pattern." This shows the background shading caused by variations in light in the original image.

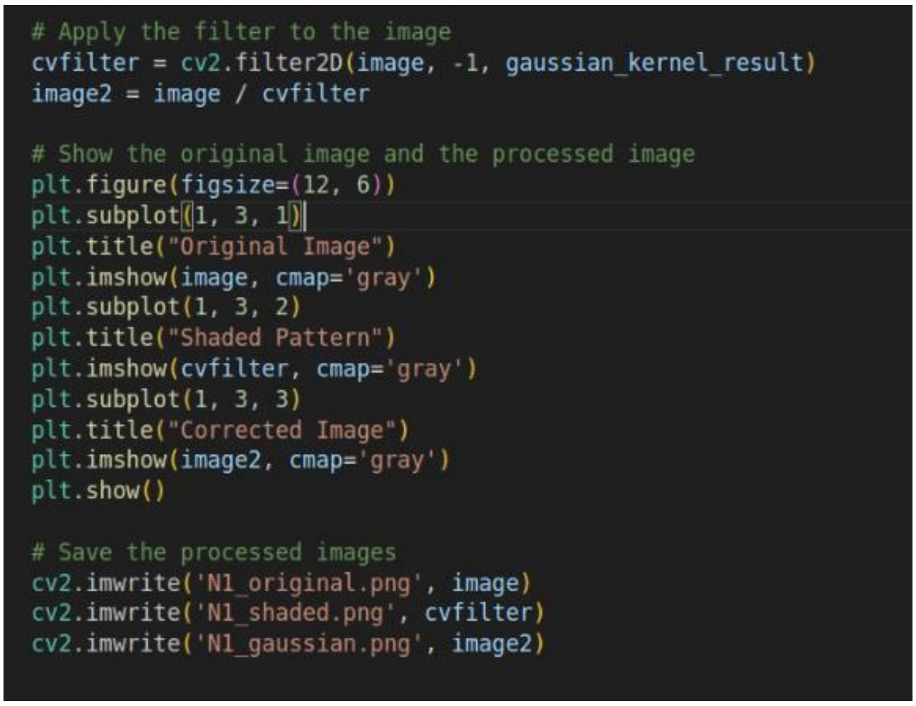
Image Correction:

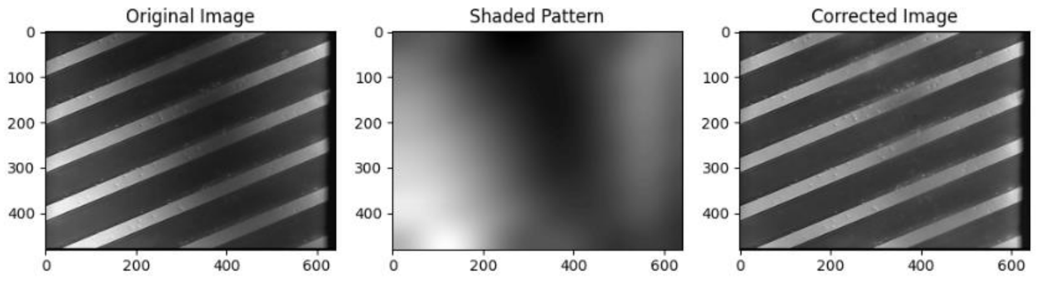
The code then divides the original image by this shading pattern to generate a "Corrected Image." The purpose of this step is to remove the shading effect from the image, making it more uniform and clearer, reducing distortions caused by uneven lighting.

Summary:

The201x201 Gaussian filter used in the code effectively smooths the image, extracts the shading pattern, and removes uneven shadows to produce a corrected image. This is a typical use of a lowpass filter, primarily for eliminating high-frequency noise and solving issues related to uneven lighting in images.


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

def gaussian_kernel(size, sigma=1):
    """
    Generate a Gaussian filter kernel
    size: The size of the kernel (should be an odd number)
    sigma: The standard deviation of the Gaussian function
    """
    # Create a grid of size x size
    ax = np.linspace(-(size // 2), size // 2, size)
    xx, yy = np.meshgrid(ax, ax)

    # Calculate the Gaussian function
    kernel = np.exp(-(xx**2 + yy**2) / (2 * sigma**2))

    # Normalize the kernel so that the sum of all values is 1
    kernel = kernel / np.sum(kernel)

    return kernel

# Set the kernel size and standard deviation
kernel_size = 201  # 121x121 kernel
sigma_value = 45

# Generate the Gaussian kernel
gaussian_kernel_result = gaussian_kernel(kernel_size, sigma_value)

# Read the image
image = cv2.imread('N1.bmp', cv2.IMREAD_GRAYSCALE)
# print(np.shape(image))
# cv2.imwrite('checkerboard.png', image)
```


```python
# Apply the filter to the image
cvfilter = cv2.filter2D(image, -1, gaussian_kernel_result)
image2 = image / cvfilter

# Show the original image and the processed image
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.title("Original Image")
plt.imshow(image, cmap='gray')
plt.subplot(1, 3, 2)
plt.title("Shaded Pattern")
plt.imshow(cvfilter, cmap='gray')
plt.subplot(1, 3, 3)
plt.title("Corrected Image")
plt.imshow(image2, cmap='gray')
plt.show()

# Save the processed images
cv2.imwrite('N1_original.png', image)
cv2.imwrite('N1_shaded.png', cvfilter)
cv2.imwrite('N1_gaussian.png', image2)
```



3. Please comment and compare your two self-designed Lowpass

Gaussian Filter Kernels? (20)

1. First Kernel

Size: 255x255

Standard Deviation (σ): 64

Characteristics:

This kernel is quite large, with a size of 255x255, meaning its influence range is very wide. A standard deviation of 64 indicates a stronger smoothing effect, resulting in more noticeable blurring across the image.

Due to the large size of the kernel, it will remove high-frequency details and noise from the image more aggressively. It is suitable for handling large-scale shading or uneven lighting conditions. It works well for scenarios that require significant blurring.

2. Second Kernel

Size: 201x201

Standard Deviation (σ): 45

Characteristics:

This kernel is smaller, at 201x201, with a lower standard deviation of 45. This results in a milder smoothing effect, meaning it will blur the image less.

This kernel is more suitable for addressing local shading or uneven lighting without overly blurring the details of the image. It's ideal for applications where preserving more image details is important.

Comparison and Commentary:

Kernel Size: The first kernel is significantly larger than the second, meaning it provides a stronger smoothing effect over a wider area, ideal for dealing with large-scale uneven lighting. In contrast, the second kernel, being smaller, is better suited for handling local detail.

Standard Deviation: The first kernel, with a higher σ (64), produces stronger blurring. The second kernel, with a lower σ (45), provides milder smoothing while retaining more detail.

Use Case: The first kernel is more appropriate for correcting large-scale lighting inconsistencies or managing large shadow patterns, while the second kernel is more appropriate for handling smaller-scale shadows or lighting variations while maintaining more image detail.

In summary, these two kernels differ in smoothing strength and intended use, allowing you to choose the most appropriate one based on your specific image processing needs.