1. Please use a Histogram Statistics method and a local enhancement method to extract the hidden image of the image, ‘hidden object.jpg.’ Please describe the your parameters and method in detail and print out the source code? (40+40)

In this program, I used a histogram statistics method and a local enhancement method to extract the hidden image from "hidden_object.jpg." Below is a detailed explanation of the parameters and methods used:

1. Histogram Statistics Method:

The purpose of the histogram statistics method is to analyze the local statistical characteristics (such as mean and standard deviation) of the image, adjust the brightness value of each pixel, and highlight hidden information within the image.

Parameter Description:

kernal_size: In the compute_stats function, the kernel size is set for local calculation. In this program, a 3x3 matrix (kernal_size=3) is used, which means the mean and standard deviation of the 3x3 neighborhood around each pixel are calculated.

k_const: This is a list containing four constant values [0, 0.3, 0, 0.04] used to control the boundary calculation for the mean and standard deviation. These constants help define the upper and lower limits when adjusting pixel values.

boundary: The calc_boundaries function generates local boundary limits based on the global mean and standard deviation of the image. These boundaries are used to determine whether each pixel should be adjusted.

adjust_histogram: For each pixel in the image, its local mean and standard deviation are checked against the calculated boundaries. If the conditions are met, the pixel's brightness value is adjusted proportionally to enhance the

hidden details.

2. Local Enhancement Method:

The purpose of the local enhancement method is to further enhance the contrast of the processed image, making the hidden image's details more apparent.

Parameter Description:

clipLimit: This parameter controls the contrast limit. The default value is used here, preventing excessive contrast enhancement in some areas of the image, thus maintaining a natural balance.

tileGridSize: This refers to the grid size used by the CLAHE (Contrast Limited Adaptive Histogram Equalization) algorithm. It controls the level of enhancement within local regions. The default parameters are used to apply contrast enhancement in various sizes of local areas.

3. Method Steps:

Histogram Statistics Processing: First, the local mean and standard deviation of the image are calculated using the compute_stats function, and the pixel values are adjusted using the adjust_histogram function. This step reduces noise in the image and highlights hidden information.

Local Enhancement Processing: After the histogram adjustment, the CLAHE local enhancement method is applied to the image to further enhance its contrast, making the hidden image more distinct.

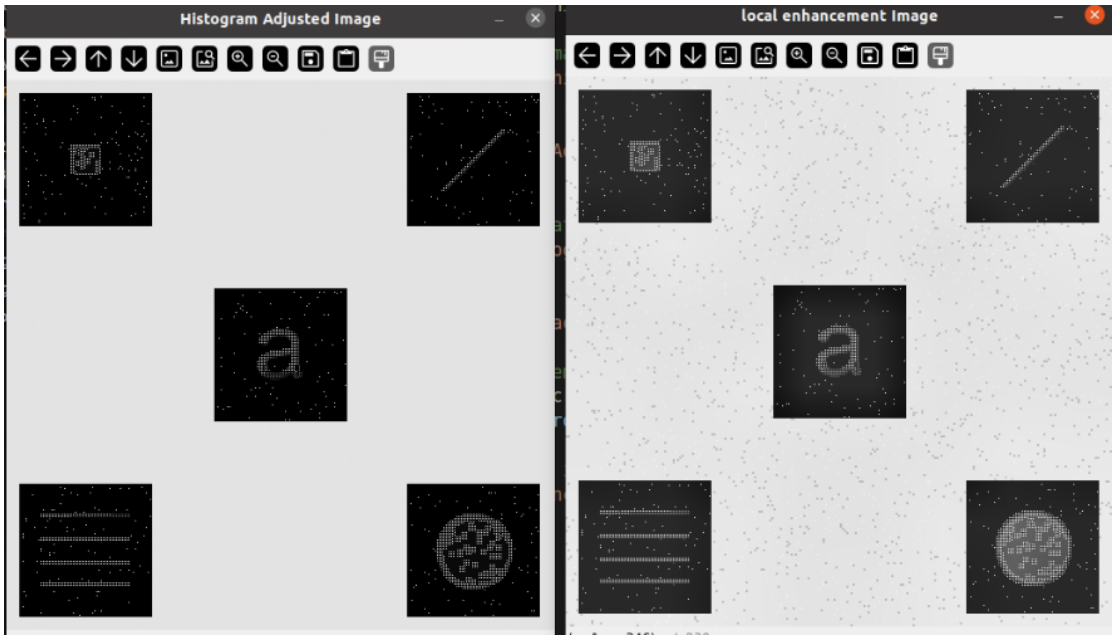
Summary:

Through these two steps, we can effectively extract the hidden image from "hidden_object.jpg." Key parameters include the kernel size for local calculations (3x3), the contrast limit, and boundary calculations based on global mean and standard deviation. This method not only removes shadows and noise from the image but also enhances the hidden details for better visibility.

```python
import cv2
import numpy as np

# Define constants
k_const = [0, 0.3, 0, 0.04]

def compute_stats(img, size):
    """
    Calculate mean and standard deviation of the image
    over a defined kernel size.
    """
    if size == 0:
        mean, std_dev = cv2.meanStdDev(img)
    else:
        kernel = np.ones((size, size), np.float32) / (size**2)
        mean = cv2.filter2D(img.astype(np.float32), -1, kernel)
        variance = cv2.filter2D((img.astype(np.float32) - mean)**2, -1, kernel)
        std_dev = np.sqrt(variance)
    return mean, std_dev

def calc_boundaries(a, b):
    """
    Calculate the boundary limits based on constants.
    """
    lower_a, upper_a = k_const[0]*a, k_const[1]*a
    lower_b, upper_b = k_const[2]*b, k_const[3]*b
    return [lower_a, upper_a, lower_b, upper_b]

def adjust_histogram(img, local_stats, max_val, bounds):
    """
    Adjust the pixel values in the image based on local statistics.
    """
    rows, cols = img.shape

    for x in range(1, rows):
        for y in range(1, cols):
            region = img[x-1:x+2, y-1:y+2]
            region_max = np.max(region)
```

```
            if bounds[0] < local_stats[0][x, y] < bounds[1] and bounds[2] < local_stats[1][x-1, y-1] < bounds[3]:
                correction_factor = round(max_val / region_max)
                img[x, y] = round(correction_factor * img[x, y])

    return img

if __name__ == '__main__':
    # Read the image
    image = cv2.imread('hidden_object_2.jpg', cv2.IMREAD_GRAYSCALE)

    # Calculate local mean and standard deviation
    local_measures = compute_stats(image, 3)

    # Find the maximum value in the image
    max_pixel_val = np.max(image)

    # Calculate the overall mean and standard deviation
    overall_mean_std = compute_stats(image, 0)

    # Calculate boundary limits
    boundaries = calc_boundaries(overall_mean_std[0], overall_mean_std[1])

    # Adjust the histogram of the image
    adjusted_img = adjust_histogram(image, local_measures, max_pixel_val, boundaries)

    # Save the resulting image
    cv2.imwrite('adjusted_histogram_image.png', adjusted_img)

    # Display the image
    cv2.imshow('Histogram Adjusted Image', adjusted_img)
    cv2.waitKey(0)

    # Save the processed data
    np.save('adjusted_histogram_data.npy', adjusted_img)
```

```
# Load the saved data
loaded_img = np.load('adjusted_histogram_data.npy')

# Perform local enhancement
clahe_processor = cv2.createCLAHE()
enhanced_img = clahe_processor.apply(loaded_img)

# Display the enhanced image
cv2.imshow('local enhancement Image', enhanced_img)
cv2.waitKey(0)
```



Histogram Statistics method local enhancement method

2. Please comment and compare Histogram Statistics method and a local enhancement method? (20)

The Histogram Statistics method and the Local Enhancement method are two common techniques in image processing, used for enhancing images and extracting details. They have different applications and effects when it comes to image enhancement.

1. Histogram Statistics Method

The Histogram Statistics method primarily analyzes the grayscale histogram of an image to adjust its contrast or brightness. A typical example is histogram equalization, which aims to distribute the grayscale values more uniformly, thus enhancing the image's contrast. This method helps in revealing details in darker or brighter areas of an image.

- **Advantages**:
    - Effective for global contrast enhancement, especially for images with even lighting.

    - Simple algorithm with relatively low computational complexity and easy to implement.

- **Disadvantages**:
    - Since it applies a global operation, it might lose some local details.
    - For images with uneven lighting, the enhancement effect may be suboptimal, leading to overexposure or overly dark regions.

2. Local Enhancement Method

The Local Enhancement method, on the other hand, processes different local regions of the image separately to enhance local details or improve contrast. Techniques like Adaptive Histogram Equalization (AHE) or local contrast enhancement fall under this category. These methods consider local characteristics of the image rather than applying a uniform adjustment to the entire image.

- **Advantages**:
    - Particularly effective for images with uneven lighting, with more noticeable enhancement of contrast and details.
    - Less likely to produce overexposed or overly dark areas, preserving more image details.
- **Disadvantages**:
    - Higher computational complexity, leading to longer processing times compared to global methods.
    - May introduce noise, especially in areas with complex textures, where enhancement might amplify the noise.

Comparison and Conclusion

The Histogram Statistics method focuses on global contrast adjustment, making it suitable for images with even lighting and offering simplicity in implementation. However, it may not perform well in images with uneven lighting. The Local Enhancement method optimizes individual regions of the

image, making it better for handling complex lighting conditions and enhancing local contrast and details, but it comes at the cost of higher computational complexity and the potential to introduce noise. Both methods have their pros and cons, and the choice of which to use depends on the characteristics of the image and the specific application requirements.