In [1]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from numpy.fft import fft2, ifft2, fftshift, ifftshift
```

Homework 2

Student id: U202115980 杨筠松 提高2101班

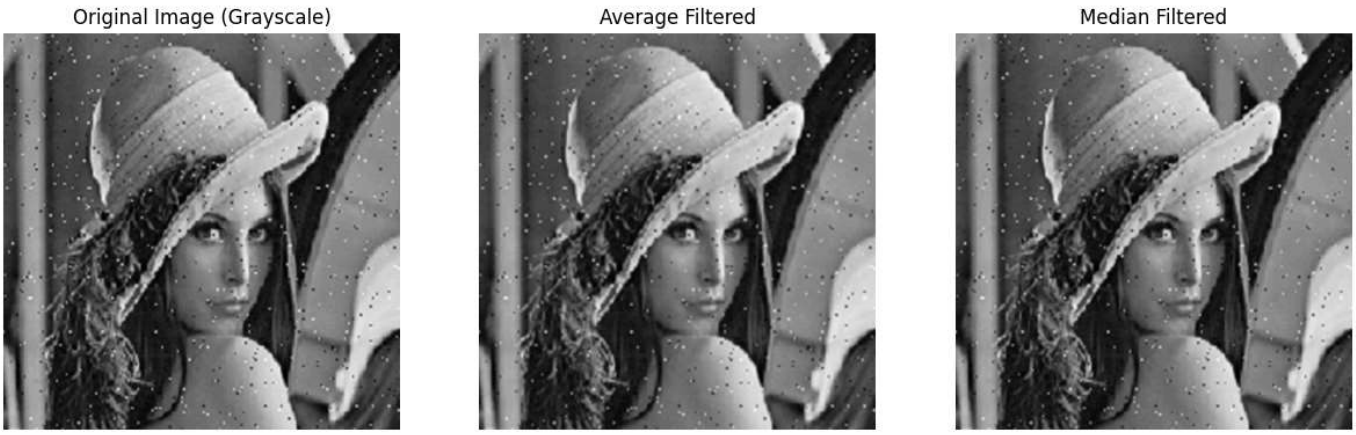
Task 1 空域滤波

In [2]:
```
# Construct spacial median filter function
def median_filter(image):
    if len(image.shape) != 2:
        raise ValueError("The median filter function expects a grayscale image")

    rows, cols = image.shape

    filtered_image = np.zeros((rows, cols), dtype=np.uint8)

    # Iterate over each pixel except the border pixels
    for i in range(1, rows - 1):
        for j in range(1, cols - 1):
            # Extract the 3x3 neighborhood
            neighborhood = image[i - 1:i + 2, j - 1:j + 2]

            # Sort the values and find the median
            median_value = np.median(neighborhood)

            # Replace the current pixel value with the median value
            filtered_image[i, j] = median_value

    filtered_image[0, :] = image[0, :]  # Top row
    filtered_image[:, 0] = image[:, 0]  # First column
    filtered_image[-1, :] = image[-1, :]  # Bottom row
    filtered_image[:, -1] = image[:, -1]  # Last column

    return filtered_image
```

In [3]:
```
# Load the image
image = cv2.imread('data/t1.png')

# Convert the image from BGR to RGB
image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Construct the filter core in 3 * 3 adjacent average core
core_avg = np.array([[1, 1, 1],
                     [1, 1, 1],
                     [1, 1, 1]]) / 9
image_avg = cv2.filter2D(image, -1, core_avg)

# Construct the filter core in 3 * 3 adjacent median core
image_median = median_filter(image)
```

In [4]:
```
# Display the image using matplotlib
fig, axs = plt.subplots(1, 3, figsize=(15, 5))
axs[0].imshow(image, cmap='gray')
axs[0].axis('off')
axs[0].set_title('Original Image (Grayscale)')
```

```python
axs[1].imshow(image_avg, cmap='gray')
axs[1].axis('off')
axs[1].set_title('Average Filtered')

axs[2].imshow(image_median, cmap='gray')
axs[2].axis('off')
axs[2].set_title('Median Filtered')

plt.show()
```

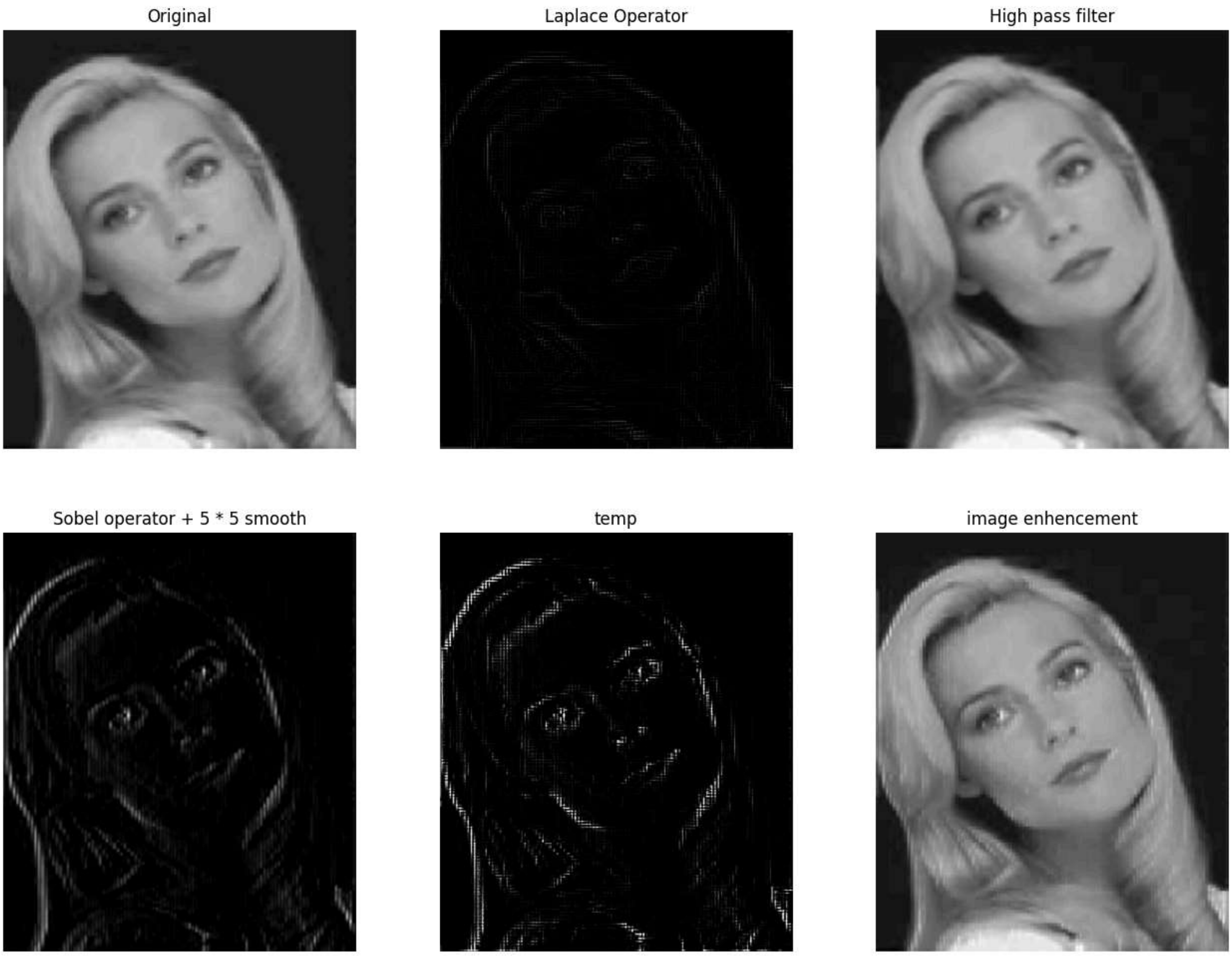


Task2 简单线性组合空域滤波

In [5]:
```python
# Load the image
image2 = cv2.imread('data/t2.png', cv2.IMREAD_GRAYSCALE)

# (a) Convert the image from BGR to RGB
# image2 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY)

# (b) Construct the filter core in laplace equation
core_laplace = np.array([[1, 1, 1],
                         [1, -8, 1],
                         [1, 1, 1]])
image_lap = cv2.filter2D(image2, -1, core_laplace)

# (c) high pass filter effect
image_hpf = cv2.add(image2, image_lap)

# (d) sobel operator + 5 * 5 smooth
core_sobelx = np.array([[-1, 0, 1],
                        [-2, 0, 2],
                        [-1, 0, 1]])
core_sobely = np.array([[-1, 2, -1],
                        [0, 0, 0],
                        [-1, 2, -1]])
core_smooth = np.ones((5, 5), np.float32) / 25
img_tempx = cv2.filter2D(image2, -1, core_sobelx)
img_tempy = cv2.add(img_tempx, cv2.filter2D(image2, -1, core_sobely))
img_sobel_smooth = cv2.filter2D(img_tempy, -1, core_smooth)

# (e) multiply (b) with (d)
img_multipication = cv2.multiply(image_lap, img_sobel_smooth)

# (f) add (e) with (a)
img_final = cv2.add(img_sobel_smooth, image2)
```

In [6]:
```python
# Display the image using matplotlib
fig, axs = plt.subplots(2, 3, figsize=(16, 12))
axs[0][0].imshow(image2, cmap='gray')
axs[0][0].axis('off')
axs[0][0].set_title('Original')
```

```python
axs[0][1].imshow(image_lap, cmap='gray')
axs[0][1].axis('off')
axs[0][1].set_title('Laplace Operator')

axs[0][2].imshow(image_hpf, cmap='gray')
axs[0][2].axis('off')
axs[0][2].set_title('High pass filter')

axs[1][0].imshow(img_sobel_smooth , cmap='gray')
axs[1][0].axis('off')
axs[1][0].set_title('Sobel operator + 5 * 5 smooth')

axs[1][1].imshow(img_multipication , cmap='gray')
axs[1][1].axis('off')
axs[1][1].set_title('temp')

axs[1][2].imshow(img_final , cmap='gray')
axs[1][2].axis('off')
axs[1][2].set_title('image enhencement')

plt.show()
```



Task3 频率滤波

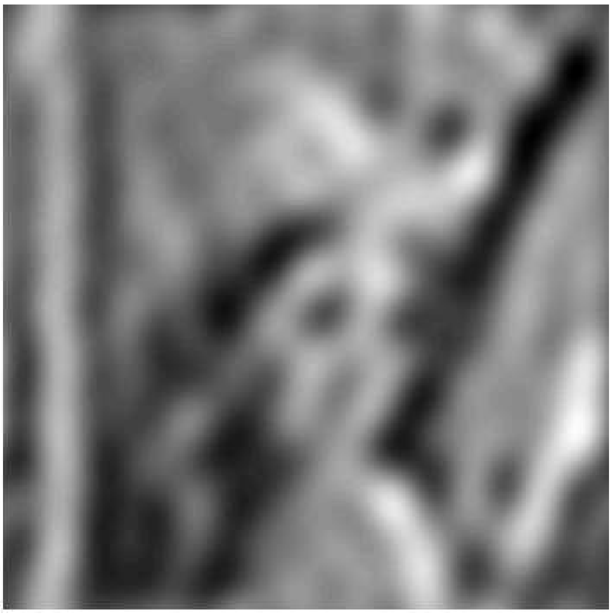
In [7]:
```python
def frequency_filter(image, radius):
    # read image from directories and calculate its fft
    dft = fft2(image)
    dft_shift = fftshift(dft)

    # calculate center
    rows, cols = image.shape
    mask = np.zeros((rows, cols), np.uint8)
```

```python
    center_row, center_col = int(rows / 2), int(cols / 2)

    for i in range(rows):
        for j in range(cols):
            if (i - center_row) ** 2 + (j - center_col) ** 2 <= radius ** 2:
                mask[i, j] = 1

    filtered_dft_shift = dft_shift * mask
    idft_shift = ifftshift(filtered_dft_shift)
    img_back = ifft2(idft_shift).real

    # Normalize the image to 8-bit range
    img_back = np.uint8(img_back / np.max(img_back) * 255)

    return img_back
```

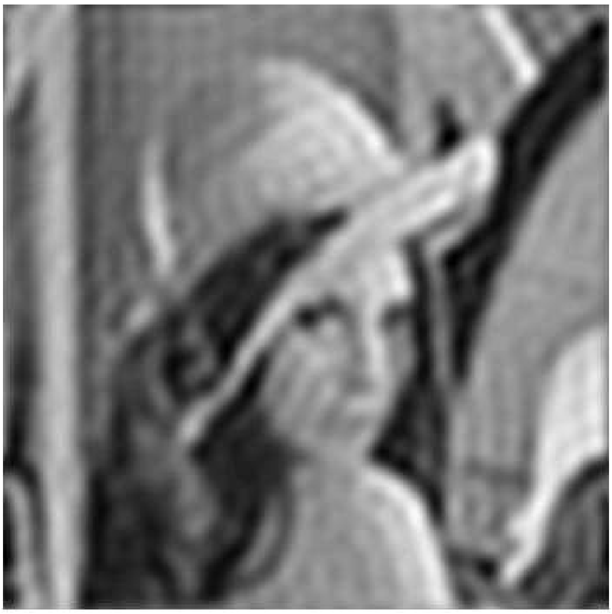
In [8]:

```python
# read image from directories and calculate its fft
image = cv2.imread('data/t3.png', cv2.IMREAD_GRAYSCALE)

rs = [10, 22, 39, 64]

# Plotting the filtered images in a 2x2 grid
fig, axs = plt.subplots(2, 2, figsize=(10, 10))

# Counter for the subplot index
counter = 0

for r in rs:
    # Apply frequency filter
    filtered_image = frequency_filter(image, r)

    # Add subplot for each radius
    axs[counter // 2, counter % 2].imshow(filtered_image, cmap='gray')
    axs[counter // 2, counter % 2].set_title(f'Radius: {r}')
    axs[counter // 2, counter % 2].axis('off')

    counter += 1

# Display the plots
plt.show()
```

Radius: 10



Radius: 22



Radius: 39

Radius: 64

Task4 美容处理-去皱

In [9]:
```
def gaussian_low_pass_filter(image, sigma):
    # Perform the FFT
    dft = fft2(image)
    dft_shift = fftshift(dft)

    # Create a Gaussian mask
    rows, cols = image.shape
    center_row, center_col = int(rows / 2), int(cols / 2)
    x, y = np.meshgrid(np.linspace(-center_col, center_col, cols),
                       np.linspace(-center_row, center_row, rows))
    gaussian_mask = np.exp(-((x**2 + y**2) / (2 * sigma**2)))

    # Apply the mask to the shifted FFT
    filtered_dft_shift = dft_shift * gaussian_mask

    # Perform the inverse FFT
    idft_shift = ifftshift(filtered_dft_shift)
    img_back = ifft2(idft_shift).real

    # Normalize the image to 8-bit range
```

```python
    img_back = np.uint8(img_back / np.max(img_back) * 255)

    return img_back
```

In [10]:
```python
# Read the image from directory
image4 = cv2.imread('data/t4.png', cv2.IMREAD_GRAYSCALE)

# Apply the Gaussian Low Pass Filter with σ = 63.2456
img_final = gaussian_low_pass_filter(image4, 63.2456)

# Set up the 1x2 subplot layout
fig, axes = plt.subplots(1, 2, figsize=(10, 5))

# Original Image
axes[0].imshow(image4, cmap='gray')
axes[0].set_title('Original Image')
axes[0].axis('off')  # Remove axes

# Filtered Image
axes[1].imshow(img_final, cmap='gray')
axes[1].set_title('Gaussian Low Pass Filtered')
axes[1].axis('off')  # Remove axes

# Display the plots
plt.tight_layout()
plt.show()
```