How to Apply Inverse Filtering, Wiener Filtering, and R-L Iterative Filtering, and Their Comparative Analysis

Introduction：

Deconvolution is a method employed to eliminate fluorescence outside the focal plane. In this field, the most classical approach is the linear inverse filtering algorithm, which posits that the deconvolution operation of the Point Spread Function (PSF) in the spatial domain is equivalent to division in the Fourier domain. However, due to sensitivity to noise and uncertainties in the system's transfer function, the inverse filtering method often yields suboptimal results, as observed in our experiments.

Wiener filtering, on the other hand, is another widely used technique in signal and image processing, aiming to restore or enhance signals or images corrupted by additive noise. Wiener filtering assumes that the observed signal is a superposition of the original signal and additive noise. When comparing inverse filtering to Wiener filtering, it becomes evident that as the noise power tends toward zero, the Wiener filter converges towards the inverse filter. While the inverse filter directly divides the observed signal by the spectrum of the original signal, Wiener filtering offers a more robust filtering effect in the presence of noise.

R-L (Richardson-Lucy) iteration is an iterative algorithm employed for image restoration and deconvolution purposes, aiming to improve the quality of images affected by blur or noise. The core concept of R-L iteration involves progressively refining the estimation of the original image through multiple iterations to minimize the impact of blur and noise, thus enhancing the image quality.

Objectives：

Application of Inverse Filtering, Wiener Filtering, and R-L Iterative Filtering Methods for Image Filtering to Remove Excess Fluorescence.

Principle：

1. Inverse Filtering Method

To begin with the inverse filtering method, it is essential to comprehend the origins of blurring and obtain the system's transfer function. This represents a crucial aspect of inverse filtering, as it elucidates how blurring affects the original image. The procedure involves performing a Fourier transform on both the observed image and the transfer function, thus transforming them into the frequency domain. Subsequently, in the frequency domain, the inverse filtering operation is executed by dividing the observed image by the spectrum of the transfer function. Finally, the inverse filtering result undergoes an inverse Fourier transform to revert the image back to the spatial domain.

$$\bar{F}(u,v) = F(u,v) + \frac{N(u,v)}{H(u,v)} \qquad 1$$

$$\bar{f}(x,y,z) = \mathcal{F}^{-1}\left\{\frac{G(\omega_x,\omega_y,\omega_z)}{H(\omega_x,\omega_y,\omega_z)}\right\} \qquad 2$$

2. Wiener Filtering Method

Wiener filtering, based on the Wiener-Kolmogorov filtering theorem, aims to estimate the spectrum of the original signal by minimizing the mean squared error, thus reducing the impact of noise. Wiener filtering operates in the frequency domain by multiplying the spectrum of the signal with an appropriate filter spectrum, effectively mitigating the influence of noise and enhancing the quality of signals or images affected by noise. The fundamental concept behind this method is the minimization of mean squared error to estimate the spectrum of the original signal, thereby minimizing the influence of noise as much as possible.

$$\hat{H}(\omega_x,\omega_y,\omega_z) = \frac{H^*(\omega_x,\omega_y,\omega_z)}{\left(|H(\omega_x,\omega_y,\omega_z)|^2 + \boxed{\frac{S_\eta(\omega_x,\omega_y,\omega_z)}{S_f(\omega_x,\omega_y,\omega_z)}}\right)} \qquad 3$$

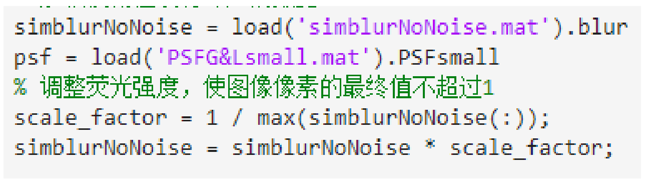
3. R-L Filtering Method

Firstly, the algorithm requires an initial estimated image, typically using a blurred or noise-corrupted observed image as the starting point. The iterative process of R-L (Richardson-Lucy) iteration comprises multiple iterative steps, with each step updating the estimated image. In each iteration step, it performs the following operations: based on the current estimated image and the model of the blur system, it calculates an estimated observed image (blurred image). The difference (residual) between the actual observed image and the estimated observed image is used as weights to update the estimated original image. This updating step aims to gradually reduce the residual through iterative refinement, thereby progressively improving the estimated image.

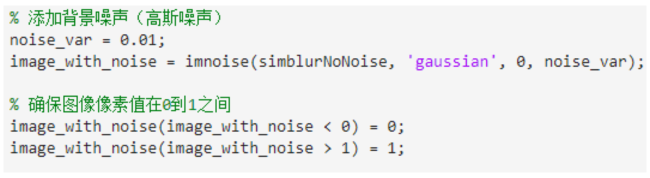
Tasks:

1. Task 1

In this step, it is necessary to perform scaling and noise addition to the computed raw data files. We employ the "imnoise" function to introduce Gaussian noise. The advantage of this operation lies in its convenience for subsequently altering the noise type and obtaining the noise variance.



```
simblurNoNoise = load('simblurNoNoise.mat').blur
psf = load('PSFG&Lsmall.mat').PSFsmall
% 调整荧光强度，使图像像素的最终值不超过1
scale_factor = 1 / max(simblurNoNoise(:));
simblurNoNoise = simblurNoNoise * scale_factor;
```



```
% 添加背景噪声（高斯噪声）
noise_var = 0.01;
image_with_noise = imnoise(simblurNoNoise, 'gaussian', 0, noise_var);

% 确保图像像素值在0到1之间
image_with_noise(image_with_noise < 0) = 0;
image_with_noise(image_with_noise > 1) = 1;
```

As the data we have obtained is three-dimensional, we have chosen to visualize the slice with an index of 25 to observe the changes. The following images depict the blurred slice before noise addition and the slice after noise has been added, sequentially.

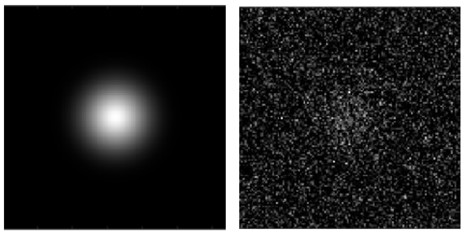
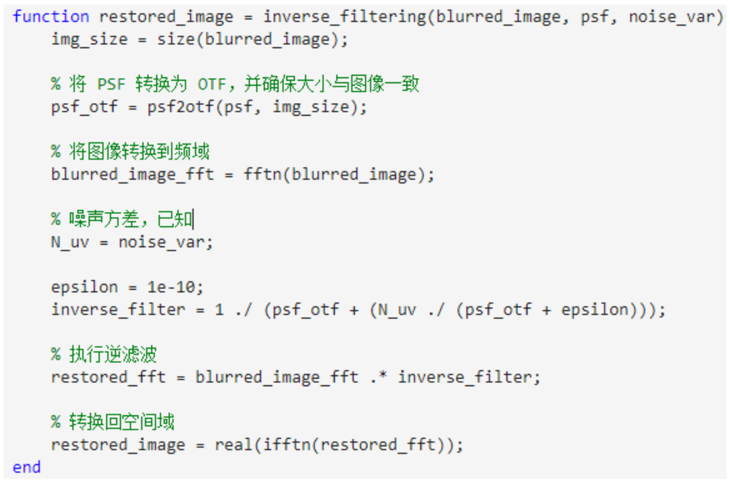


Fig.1 the blurred slice before noise nd the slice after noise

2. Task 2

We have completed the inverse filtering method, and we possess knowledge of the point spread function and the noisy data. Therefore, following the formula for inverse filtering, we have matched the parameters correspondingly.



```
function restored_image = inverse_filtering(blurred_image, psf, noise_var)
    img_size = size(blurred_image);

    % 将 PSF 转换为 OTF，并确保大小与图像一致
    psf_otf = psf2otf(psf, img_size);

    % 将图像转换到频域
    blurred_image_fft = fftn(blurred_image);

    % 噪声方差，已知
    N_uv = noise_var;

    epsilon = 1e-10;
    inverse_filter = 1 ./ (psf_otf + (N_uv ./ (psf_otf + epsilon)));

    % 执行逆滤波
    restored_fft = blurred_image_fft .* inverse_filter;

    % 转换回空间域
    restored_image = real(ifftn(restored_fft));
end
```

After running the process, we have obtained the following filtering results:

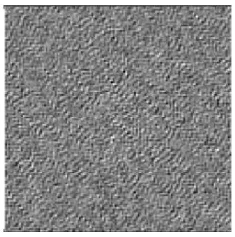
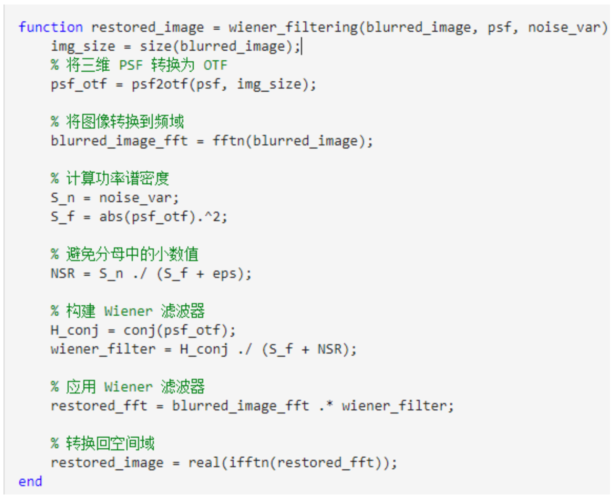


Fig.2 Inverse Filtering Method

We have observed that the results are not satisfactory, which may be attributed to the inherent sensitivity of this method to noise, thereby amplifying the influence of the noise.

Continuing with the Wiener filtering process, Wiener filtering requires the estimation of the power spectral density functions of both the signal and the noise. Power spectral density describes the frequency-domain characteristics of the signal and noise and is typically estimated by performing Fourier transforms on sample data of the signal and noise. The Wiener filtering function is a frequency-domain filter designed to minimize the mean square error between the output signal and the original signal. This filter function is often denoted as H(f), where f represents frequency. The computation of the Wiener filtering function involves the power spectral density functions of the signal and noise, as well as the correlation between the signal and noise.



```matlab
function restored_image = wiener_filtering(blurred_image, psf, noise_var)
    img_size = size(blurred_image);
    % 将三维 PSF 转换为 OTF
    psf_otf = psf2otf(psf, img_size);

    % 将图像转换到频域
    blurred_image_fft = fftn(blurred_image);

    % 计算功率谱密度
    S_n = noise_var;
    S_f = abs(psf_otf).^2;

    % 避免分母中的小数值
    NSR = S_n ./ (S_f + eps);

    % 构建 Wiener 滤波器
    H_conj = conj(psf_otf);
    wiener_filter = H_conj ./ (S_f + NSR);

    % 应用 Wiener 滤波器
    restored_fft = blurred_image_fft .* wiener_filter;

    % 转换回空间域
    restored_image = real(ifftn(restored_fft));
end
```

We obtained the slice with an index of 4, and the results can be observed as follows:

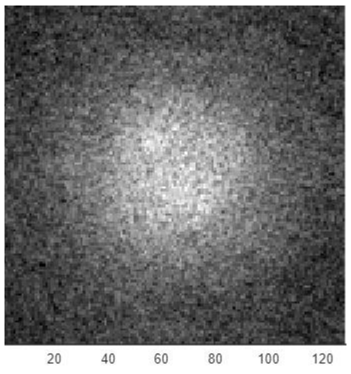
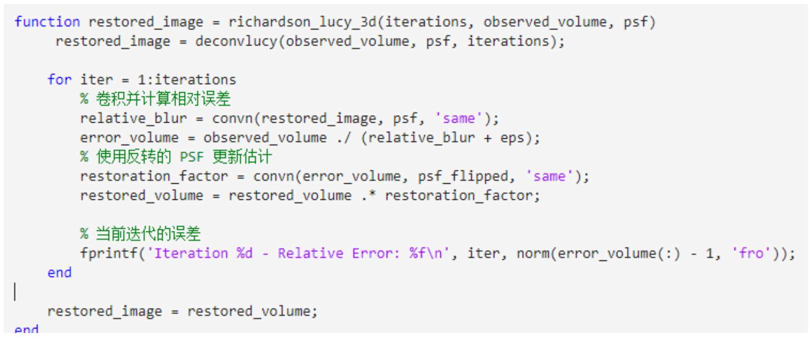


Fig.3 Wiener Filtering Method

3. Task 3

In the context of the RL (Richardson-Lucy) filtering, we primarily employ an iterative approach. Initially, we engage in the Blurred Backprojection step, which involves subjecting the current estimate of the image to a deconvolution process, followed by a comparison with the observed image, resulting in an error image. Subsequently, we proceed to the Weighted Adjustment step. In this step, we utilize the error image to adjust the weights associated with the current image estimate, aiming to improve the estimation in the next iteration.



```matlab
function restored_image = richardson_lucy_3d(iterations, observed_volume, psf)
     restored_image = deconvlucy(observed_volume, psf, iterations);

    for iter = 1:iterations
        % 卷积并计算相对误差
        relative_blur = convn(restored_image, psf, 'same');
        error_volume = observed_volume ./ (relative_blur + eps);
        % 使用反转的 PSF 更新估计
        restoration_factor = convn(error_volume, psf_flipped, 'same');
        restored_volume = restored_volume .* restoration_factor;

        % 当前迭代的误差
        fprintf('Iteration %d - Relative Error: %f\n', iter, norm(error_volume(:) - 1, 'fro'));
    end

    restored_image = restored_volume;
end
```

After undergoing iterative processing, we visualize the final results:

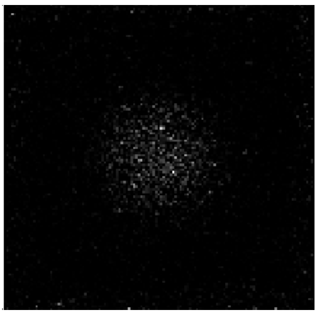


Fig.4 Richardson-Lucy filtering

We observed that our results exhibit significant errors, and the errors decrease rapidly, suggesting that our initial initialization is far from the correct outcome. The Richardson-Lucy (R-L) filtering method is tailored for Poisson distributed noise; however, the noise introduced in our case is Gaussian, which to some extent adversely impacts the results.

4. Task 4

After a comparative analysis, we have found that both the Wiener filtering method and the RL (Richardson-Lucy) filtering method yield superior results. The Wiener filtering method often employs regularization techniques, which enhance robustness when dealing with noise, preventing excessive noise amplification. Additionally, Wiener filtering allows for the adjustment of filter parameters to strike a balance between blur and noise, enabling improved results in various scenarios, especially when noise levels and blur intensities are non-uniform or variable.

On the other hand, the RL filtering method utilizes an iterative approach, iteratively updating estimates to gradually reduce the impact of blur and noise. This iterative process enables better adaptation to the details and complexity of the signal.

In summary, both Wiener and RL filtering methods are more flexible and adaptable to different situations, providing robustness and the ability to fine-tune parameters. These qualities contribute to their superior performance compared to simple inverse filtering methods. However, it is essential to exercise careful consideration when choosing the appropriate method, as these techniques may require more computational resources and parameter adjustments in practical applications.

5. Task 5

We applied the aforementioned methods to real-world data, keeping all other conditions unchanged, and subsequently observed the outcomes.

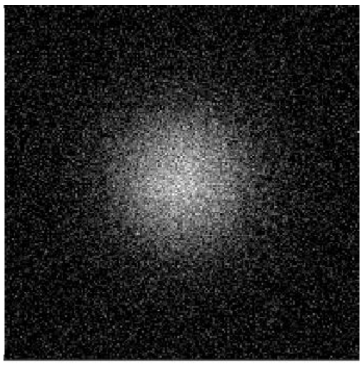


Fig.5 with noise real data

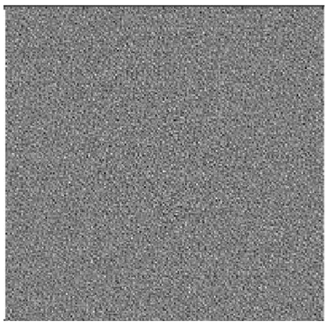


Fig.6 real data after inverse filtering

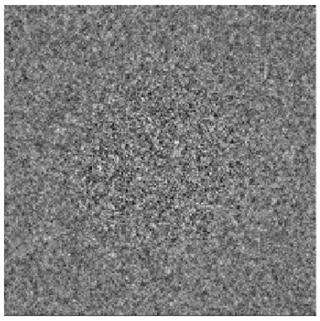


Fig.7 real data after wiener filtering

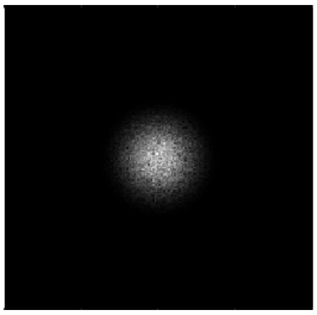


Fig.8 real data after R-L

Through comparisons conducted on actual data, we have observed that individual outcomes may exhibit variations. However, upon conducting a cross-comparison, we have found that on real-world data, we can still arrive at the same conclusions as those derived from computed data.

Discussion:

Wiener filtering typically relies on prior knowledge, such as the power spectral density functions of the signal and noise, as well as the correlation between the signal and noise. If these assumptions are inaccurate, the filtering effectiveness may be compromised. Wiener filtering is highly sensitive to noise estimation; if the noise model is inaccurate or the noise is substantial, it may not effectively remove noise or could lead to excessive noise amplification. The selection of parameters in Wiener filtering, such as the filter's cutoff frequency or regularization parameter, often requires empirical or trial-and-error approaches, and different parameter choices may yield different filtering results.

Furthermore, Wiener filtering's computational complexity is a concern. RL (Richardson-Lucy) filtering, similar to Wiener filtering, typically involves multiple iterations, which can result in high computational demands, particularly when dealing with large-scale images. A greater number of iterations can lead to longer processing times. RL filtering also necessitates an initial estimate of the image, and if the initial estimate is inaccurate, it may cause the algorithm to converge to a local minimum, resulting in unstable results.

Similarly to Wiener filtering, RL filtering is sensitive to noise, especially during the iterative process, where it may amplify noise, leading to unstable estimation results. Selecting appropriate stopping criteria for the iterations in RL filtering can be challenging, and improper choices may result in overfitting or underfitting.