Signals and Systems
Mekelweg 4,
2628 CD Delft
The Netherlands
https://sps.ewi.tudelft.nl/

SPS-2024-00

M.Sc. Thesis

3D Microscopy Deconvolution of Very Large Images with an Adaptive Resolution Scheme

Chang Ge

Abstract

Microscopy is a crucial tool across various scientific domains. Due to light diffraction, the images acquired from optical microscopes are usually blurry and corrupted by noise. For an accurate quantitative analysis, the measured images need to be deconvolved to achieve higher resolution. Deconvolution processes are computationally expensive, due to the large data size. This leads to out-of-memory issues and extended computation time.

To address these problems, this project aims to develop a novel convolution scheme. It utilizes the special structure of the Point Spread Function, which in conventional microscopy techniques has most of its energy concentrated in the center. And implement multi-resolution signal processing methods. This approach enhances computational efficiency while retaining computational accuracy.

3D Microscopy Deconvolution of Very Large Images with an Adaptive Resolution Scheme

---

Thesis

submitted in partial fulfillment of the
requirements for the degree of

Master of Science

in

Electrical Engineering

by

Chang Ge
born in Jinan, China

This work was performed in:

Signals and Systems Group
Department of Microelectronics
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology

Delft University of Technology

Delft University of Technology
Department of
Microelectronics

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled **"3D Microscopy Deconvolution of Very Large Images with an Adaptive Resolution Scheme "** by **Chang Ge** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: 18 October 2024

Chairman: _____

prof.dr.ir. Alle-Jan van der Veen

Advisor: _____

Committee Members: _____

_____

Abstract

Microscopy is a crucial tool across various scientific domains. Due to light diffraction, the images acquired from optical microscopes are usually blurry and corrupted by noise. For an accurate quantitative analysis, the measured images need to be deconvolved to achieve higher resolution. Deconvolution processes are computationally expensive, due to the large data size. This leads to out-of-memory issues and extended computation time.

To address these problems, this project aims to develop a novel convolution scheme. It utilizes the special structure of the Point Spread Function, which in conventional microscopy techniques has most of its energy concentrated in the center. And implement multi-resolution signal processing methods. This approach enhances computational efficiency while retaining computational accuracy.

Acknowledgments

I would like to express my sincere gratitude to my supervisors, Dr. Daniel Sage, Dr. Joan Rué Queralt, Dr. Vasiliki Stergiopoulou, and Prof. Alle-Jan van der Veen, for their support and guidance throughout this journey.

Chang Ge
Delft, The Netherlands
18 October 2024

Contents

List of Figures

List of Tables

1 Introduction

1.1 Microscope imaging

A microscope is an optical instrument used to magnify small objects, and it is widely employed in various fields, particularly in biological studies to observe and analyze live specimens. Among the different types of light microscopes, fluorescence microscopy has emerged as a powerful technique, largely due to advancements in fluorescent labeling methods.

Fluorescence microscopy utilizes the phenomenon of fluorescence to highlight specific structures within biological specimens. Figure 1.1 shows a schematic plot of a fluorescence microscope, it typically consists of magnifying optical lenses, optical filters, and a dichroic mirror to reflect and transmit certain wavelengths.

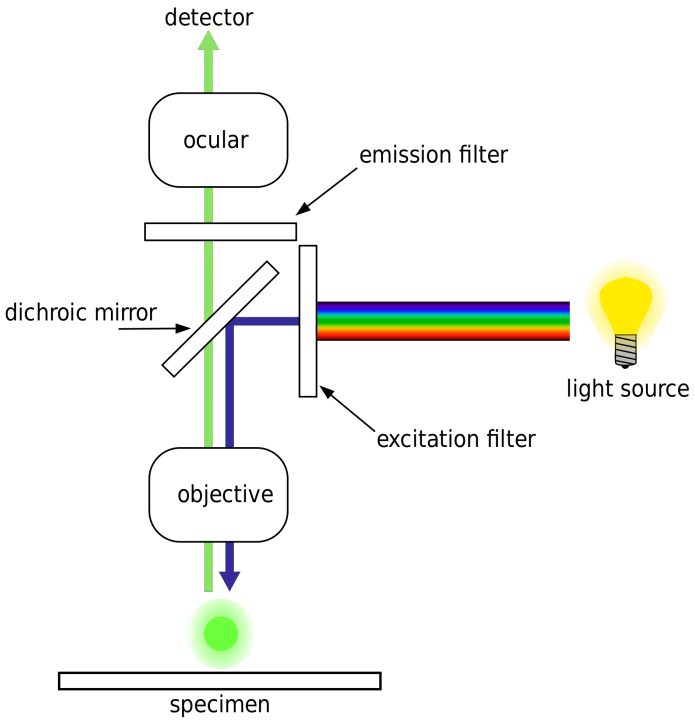


Figure 1.1: Schematic of a fluorescence microscope[1]

Biological specimens are first dyed with fluorescent dyes. Lights of specific wavelengths illuminate and excite the fluorophores in the samples, which then emit fluorescent light under another wavelength. Only the emitted light passes through the dichroic mirror

and the filter. The optical sensor detects the light and forms an image. These techniques enable detailed visualization of cellular components, providing high specificity and contrast, and have thus become widely used in biological research.

Two of the most common fluorescent microscopes are the widefield and the confocal ones. Figure 1.2 compares the different illumination schemes of widefield and confocal fluorescence microscopes. In wide-field microscopy, the entire specimen is evenly illuminated by the light source. In confocal microscopy, the light is focused on a limited spot of the specimen, sequentially illuminating different parts of the object. It contains a small pinhole in the conjugate focus plane to block out-of-focus light. This scanning method is more complex and time-consuming but contributes to improved resolution and contrast.

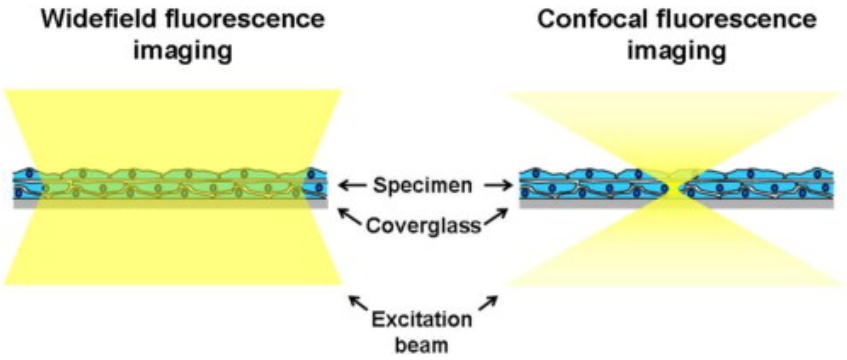


Figure 1.2: Comparison of fluorescence illumination in widefield and confocal fluorescence microscopy. [2]

As shown in Figure 1.3, by adjusting the focal plane through the sample, a series of 2D images can be obtained. These z-stack images are then compiled to form a 3D microscope image. Each 2D image is considered a projection of the 3D object along the optical axis. 3D information outside the focal plane appears as a blurry background.

1.2 Motivations for deconvolution

However, images measured with fluorescence microscopes often suffer from degradation. This becomes problematic for the quantitative analysis of the images. The image degradations are caused by optical blur and sensor shot noise [4].

The blurry effect is caused by optical diffraction. The optical blur effect is inherent to the optical system and can be characterized by the Point Spread Function (PSF). Due to the diffraction property of light waves, a single point will appear as a Point Spread Function in microscopy images. For 3D microscopy, it has the shape of a double cone.

Noise in microscopy images includes non-additive signal-dependent shot noise. It follows a Poisson distribution with the mean of the detected light. Additionally, the camera sensors bring thermal and background noise. Moreover, for each 2D image at a certain focal plane, out-of-focus information from other planes is also projected onto

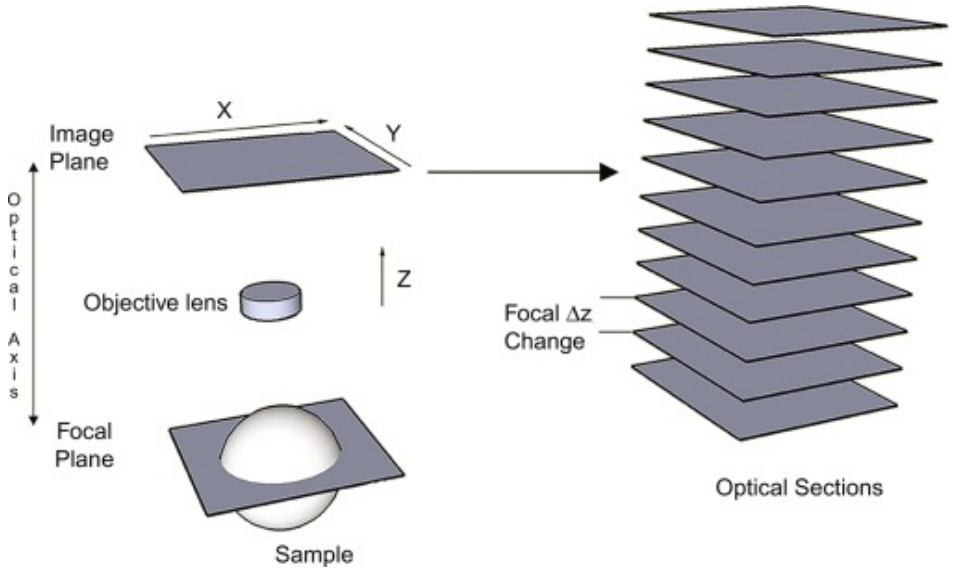


Figure 1.3: A schematic representation of 3D image formation in microscopy [3]

the same plane. This also contributes to background noise. They can be modeled as additive Gaussian noise.

To retrieve desired 3D microscope images from degraded measurement images, deconvolution methods need to be performed. Deconvolution methods typically involve iterative convolution operations with a PSF as shown in Section 2.2. However, these convolutions are computationally expensive in both time and memory due to the large data sizes of both measurement data and PSF. Iterative deconvolution algorithms such as the Richardson-Lucy algorithm are usually slow to converge. This often leads to out-of-memory issues and extended computation times. For example, convolving an image of voxels of size $128 \times 128 \times 100$ with a PSF of the same size requires 6.8 GB of memory[5]. In other cases, the size of PSF would be $1024 \times 1024 \times 100$. In extreme cases, data size can reach to $4096 \times 4096 \times 1000 = 16'777'216'000$ voxels. This will cause more computational burden. To perform deconvolution algorithms would take hours[6].

To address these issues, various methods have been proposed through accelerated deconvolution algorithms and enhanced convolution efficiency. D.S. Biggs accelerated the Richardson-Lucy algorithm by adjusting the step size, improving its convergence speed[7]. M. Guo et al. utilized an unmatched back projector to reduce the number of iterations needed, enhancing deconvolution speed by 10-100 times through CPU processing[8]. D. Svoboda reduced memory overhead and achieved faster convolutions by implementing image and kernel tiling techniques[5].

This project aims to develop a faster and more efficient method to deconvolve microscopy images. Leveraging the special structure of PSF, we design a multiscale convolution algorithm with faster computing time, lower memory overhead, and small accuracy loss.

1.3 Point Spread Function

PSF serves as the convolution kernel in the forward model and thus plays a key role in microscopy image deconvolution. With prior knowledge of the microscope, the PSF is typically assumed to be known. It can be obtained through experimental approaches or theoretical simulations [9]. In experimental approaches, images of point-like objects or beads are measured to estimate the PSF. This method better captures the actual characteristics of the optical system. However, these experimentally acquired PSF images often contain noise and have a low signal-to-noise ratio (SNR). Figure **??** shows an example of theoretical and experimental PSFs. Compared to the theoretical PSF, the experimental PSF has an irregular shape and contains noise.

There are several types of PSF models used to obtain theoretical PSFs. The Born and Wolf model is a scalar-based diffraction model that derives the PSF from the Fourier transform of the optical system's apertures [10], [11]. The Gibson and Lanni model further considers variations in the refractive index for thick specimens, allowing for a more accurate representation of the PSF in three dimensions [12]. Other PSF models simulate the blurring effect through a 2D Gaussian function, either in the Fourier domain or in the spatial domain. Software such as PSF Generator [13] can be used to generate PSFs with various simulation models based on the parameters of the optical system.

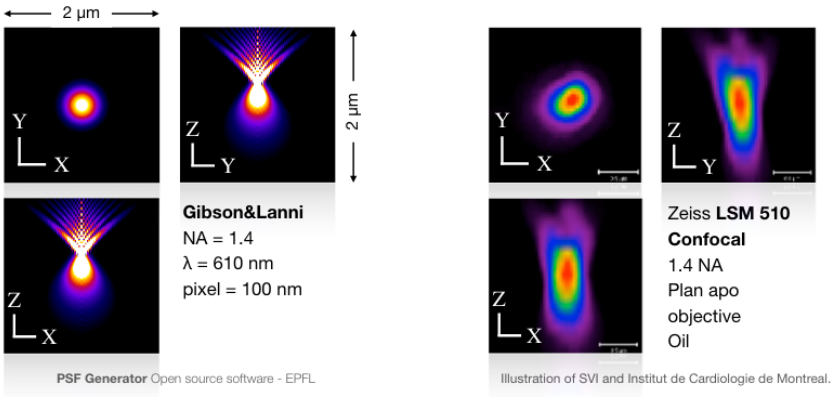


Figure 1.4: Comparison of theoretical and experimental PSF. [13]

1.4 Outline

The outline of this thesis report is as follows:

- Chapter 2: This chapter includes the forward model, the deconvolution algorithms, and convolution methods.
- Chapter 3: The proposed convolution methods will be described in detail. The computational complexity of the proposed algorithm will also be analyzed.

- Chapter 4: This chapter will present the convolution and deconvolution results using different types of data.
- Chapter 5: Discuss the limitations of the algorithm and suggest potential directions for further research.

References

[1] W. Commons, *File:fluoreszenzmikroskopie 2008-09-28.svg — wikimedia commons, the free media repository*, [Online; accessed 22-July-2024], 2023. [Online]. Available: https://commons.wikimedia.org/w/index.php?title=File:Fluoreszenzmikroskopie_2008-09-28.svg&oldid=796566148.

[2] J. Chatton, "Ionic homeostasis in glia: A fluorescence microscopy approach," in *Reference Module in Biomedical Sciences*, Elsevier, 2015, ISBN: 978-0-12-801238-3. DOI: https://doi.org/10.1016/B978-0-12-801238-3.04624-9. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128012383046249.

[3] F. A. Merchant and A. Diaspro, "Chapter eleven - three-dimensional imaging," in *Microscope Image Processing (Second Edition)*, F. A. Merchant and K. R. Castleman, Eds., Second Edition, Academic Press, 2023, pp. 247–317, ISBN: 978-0-12-821049-9. DOI: https://doi.org/10.1016/B978-0-12-821049-9.00009-5. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128210499000095.

[4] J.-B. Sibarita, "Deconvolution microscopy," *Advances in biochemical engineering/biotechnology*, vol. 95, pp. 201–43, Feb. 2005. DOI: 10.1007/b102215.

[5] D. Svoboda, "Efficient computation of convolution of huge images," in *Image Analysis and Processing–ICIAP 2011: 16th International Conference, Ravenna, Italy, September 14-16, 2011, Proceedings, Part I 16*, Springer, 2011, pp. 453–462.

[6] P. Sarder and A. Nehorai, "Deconvolution methods for 3-d fluorescence microscopy images," *IEEE signal processing magazine*, vol. 23, no. 3, pp. 32–45, 2006.

[7] D. S. Biggs and M. Andrews, "Acceleration of iterative image restoration algorithms," *Applied optics*, vol. 36, no. 8, pp. 1766–1775, 1997.

[8] M. Guo, Y. Li, Y. Su, *et al.*, "Rapid image deconvolution and multiview fusion for optical microscopy," en, *Nature Biotechnology*, vol. 38, no. 11, pp. 1337–1346, Nov. 2020, ISSN: 1087-0156, 1546-1696. DOI: 10.1038/s41587-020-0560-x. [Online]. Available: https://www.nature.com/articles/s41587-020-0560-x (visited on 05/15/2024).

[9] J. Markham and J.-A. Conchello, "Parametric blind deconvolution: A robust method for the simultaneous estimation of image and blur," *JOSA A*, vol. 16, no. 10, pp. 2377–2391, 1999.

[10] F. Aguet, D. Van De Ville, and M. Unser, "Model-based 2.5-d deconvolution for extended depth of field in brightfield microscopy," *IEEE Transactions on Image Processing*, vol. 17, no. 7, pp. 1144–1153, 2008.

[11] M. Born and E. Wolf, *Principles of optics: electromagnetic theory of propagation, interference and diffraction of light*. Elsevier, 2013.

[12] S. F. Gibson and F. Lanni, "Diffraction by a circular aperture as a model for three-dimensional optical microscopy," *JOSA A*, vol. 6, no. 9, pp. 1357–1367, 1989.

[13] H. Kirshner, D. Sage, and M. Unser, "3d psf models for fluorescence microscopy in ImageJ," in *Proceedings of the Twelfth International Conference on Methods and Applications of Fluorescence Spectroscopy, Imaging and Probes (MAF'11)*, Strasbourg French Republic, vol. 154, 2011.

2 Problem formulation

2.1 Forward model

The microscopy imaging process can be modeled as a linear shift-invariant system, where the Point Spread Function (PSF) serves as the impulse response. For simplicity, the model is first explained in 1D and later extended to the d-dimensional case, where $d \in \{1, 2, 3\}$. Figure 2.1 shows an example of the 1D microscope imaging system.

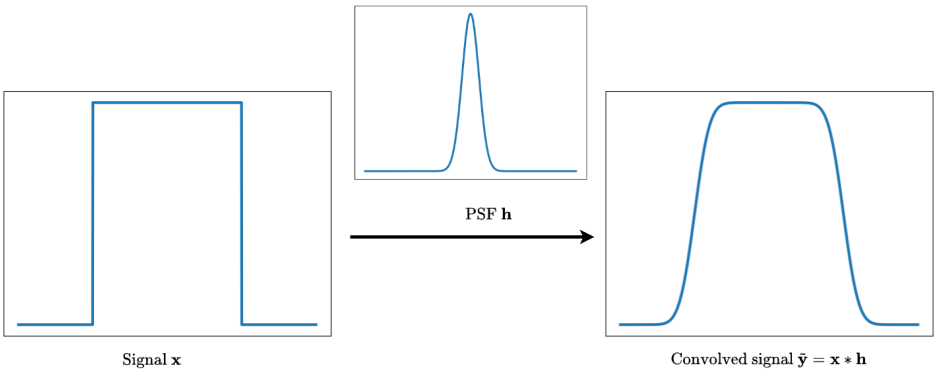


Figure 2.1: Microscope imaging system

Under the assumption of a noise-free environment, the measured signal is a blurry image, denoted as $\tilde{\mathbf{y}}$. This can be described as the target image $\mathbf{x}$ convolved with the PSF $\mathbf{h}$ of the microscope. The PSF represents the diffraction pattern of an infinitesimally small point source within the microscope. The forward model is represented as:

$$\tilde{\mathbf{y}} = \mathbf{h} * \mathbf{x},$$

where $*$ denotes the convolution operation, and $\tilde{\mathbf{y}}, \mathbf{x}, \mathbf{h} \in \mathbb{R}^N$ represent the measured blurry image vector, the target image vector, and the PSF vector, respectively. This model can also be written in matrix form as:

$$\tilde{\mathbf{y}} = \mathbf{H}\mathbf{x},$$

where $\mathbf{H}$ is the linear operator representing the convolution with the PSF. For 1D data, $\mathbf{H}$ takes the form of a circular matrix. Additionally, $\mathbf{H}$ can be expressed using the Discrete Fourier Transform (DFT) matrix $\mathbf{F}$ and its complex conjugate $\mathbf{F}^H$ as:

$$\mathbf{H} = \mathbf{F}^{\mathbf{H}}\mathbf{\Lambda}\mathbf{F}, \quad \text{where} \quad \mathbf{\Lambda} = \text{diag}(\mathbf{F}\mathbf{h}).$$

Here, $\mathbf{H}$ is of size $N \times N$, which makes it impractical to store directly in memory. Instead, $\mathbf{H}$ is often implemented in a matrix-free manner, where its elements are computed as needed during operations.

Performing this convolution with microscope data is computationally expensive due to the large size of both the image data and PSF. With the image size proposed above, the direct convolution method involves computing each output element as a sum of element-wise products of the input signal and the kernel. This requires N multiplications and N additions for each of the N output elements, leading to a total computational complexity of $\mathcal{O}(N^2)$.

Fourier-based convolution includes transforming the input signal and kernel to the frequency domain using Fast Fourier Transform (FFT), performing element-wise multiplication in the frequency domain, and then transforming the result back to the spatial domain using inverse FFT. FFT transform has a complexity of $\mathcal{O}(N \log N)$, and the element-wise multiplication has a complexity of $\mathcal{O}(N)$, leading to an overall computational complexity of $\mathcal{O}(N \log N)$. For large N, the FFT-based method is generally more efficient than the direct method due to its lower computational complexity.

The measurement model considering the shot noise and additive Gaussian noise is illustrated as follows. The intensity of shot noise depends on the light intensity and is non-additive. It can be modeled using an independent Poisson process [1], [2]. Denote $\mathcal{P}(\cdot)$ as the Poisson distribution. Additionally, $\mathbf{n}$ denotes independent and identically distributed (i.i.d.) Gaussian noise.

Thus, the measurement model can be written as:

$$\mathbf{y} = \mathcal{P}(\mathbf{Hx}) + \mathbf{n}$$

2.2 Deconvolution methods

To recover $\mathbf{x}$ from noisy measurement $\mathbf{y}$, deconvolution algorithms can be applied. They can be characterized into direct inversion and iterative deconvolution methods.

2.2.1 Direct inversion

Direct inversion aims to recover the original signal $\mathbf{x}$ from the measured signal $\mathbf{y}$ by inverting the PSF in the Fourier domain. By the convolution theorem, the forward model is equal to multiplying $\mathbf{h}$ and $\mathbf{x}$ in Fourier domain, thus the inversion can be implemented through division. This is mathematically expressed as:

$$\hat{\mathbf{x}}[k] = \begin{cases} \mathbf{F}^{-1}\left(\frac{\mathbf{Fy}}{\mathbf{Fh}}\right)[k] & \text{if } \mathbf{Fh}[k] \geq \epsilon \\ 0 & \text{if } \mathbf{Fh}[k] < \epsilon \end{cases}.$$

Here, $\hat{\mathbf{x}}$ represents the estimated signal in the spatial domain, while $\mathbf{F}$ and $\mathbf{F}^{-1}$ denote the Discrete Fourier Transform (DFT) matrix and its inverse, respectively. The

term $\mathbf{Fy}$ refers to the DFT of the measured signal $\mathbf{y}$, and $\mathbf{Fh}$ represents the DFT of the point spread function (PSF) $\mathbf{h}$. The element-wise division $\frac{\mathbf{Fy}}{\mathbf{Fh}}$ is used to perform the deconvolution in the frequency domain. The threshold ϵ is introduced to prevent division by values that are close to zero.

Although this method is fast and does not require any additional parameters, it performs poorly in practice when applied to noisy measurements, particularly in the presence of dominant Poisson noise. Direct inversion in such cases often results in significant noise amplification, which in turn degrades the quality of the estimated signal $\hat{\mathbf{x}}$.

2.2.2 Iterative deconvolution

One of the widely used iterative deconvolution methods is the Richardson-Lucy algorithm [2]–[4]. It is a maximum likelihood algorithm based on the Poisson noise model, which makes it well-suited for our purposes, as microscopy data are mainly corrupted by Poisson noise. It assumes that there is no additive Gaussian noise, which leads to the measurement model:

$$\mathbf{y} = \mathcal{P}(\mathbf{Hx}), \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^N.$$

Let $\mathbf{x}[i]$, $\mathbf{y}[i]$ denote the i-th elements of $\mathbf{x}$ and $\mathbf{y}$, respectively, for $i = 1, 2, \ldots, N$. The Poisson likelihood function is given by:

$$p(\mathbf{y} \mid \mathbf{Hx}) = \prod_{i=1}^{N} \frac{((\mathbf{Hx})[i])^{\mathbf{y}[i]} e^{-(\mathbf{Hx})[i]}}{\mathbf{y}[i]!}$$

Define $\times$ as element-wise multiplication, $\mathbf{H}^T$ is the adjoint operator of $\mathbf{H}$ and can be viewed as the convolution with reversed PSF $\mathbf{h}$ and $\mathbf{1}$ is an N by 1 all one vector. Thus we can write:

$$p(\mathbf{y} \mid \mathbf{Hx}) = \exp\left(-(\mathbf{Hx})^\top \mathbf{1}\right) \cdot \exp\left(\log(\mathbf{Hx})^\top \mathbf{y}\right) \cdot \prod_{i=1}^{N} \frac{1}{\mathbf{y}[i]!}.$$

Taking the log-likelihood function of $\mathbf{y}$, we can write:

$$L(\mathbf{x}) = \log(p(\mathbf{y} \mid \mathbf{Hx})) = -(\mathbf{Hx})^\top \mathbf{1} + \log(\mathbf{Hx})^\top \mathbf{y} - \sum_{i=1}^{N} \log(\mathbf{y}[i]!).$$

The gradient of the log-likelihood function with respect to $\mathbf{x}$ is :

$$\nabla L(\mathbf{x}) = \mathbf{H}^\top \frac{\mathbf{y}}{\mathbf{Hx}} - \mathbf{H}^\top \mathbf{1},$$

where $\frac{\mathbf{y}}{\mathbf{Hx}}$ is an element-wise division. Let $\mathbf{x}^k$ represent the k-th iteration of the algorithm. Then an iterative update rule can be formulated using a gradient ascent approach to maximize the log-likelihood:

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \lambda \nabla L(\mathbf{x}^k).$$

The step size λ is chosen such that: $\lambda = \frac{\mathbf{x}^k}{\mathbf{H}^T\mathbf{1}}$. This leads to an iterative update rule:

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathbf{x}^k + \frac{\mathbf{x}^k}{\mathbf{H}^\top\mathbf{1}} \times \left(\mathbf{H}^\top \frac{\mathbf{y}}{\mathbf{H}\mathbf{x}^k} - \mathbf{H}^\top\mathbf{1}\right) \\ &= \mathbf{x}^k + \frac{\mathbf{x}^k}{\mathbf{H}^\top\mathbf{1}} \times \mathbf{H}^\top\left(\frac{\mathbf{y}}{\mathbf{H}\mathbf{x}^k}\right) - \mathbf{x}^k \\ &= \frac{\mathbf{x}^k}{\mathbf{H}^\top\mathbf{1}} \times \mathbf{H}^\top\left(\frac{\mathbf{y}}{\mathbf{H}\mathbf{x}^k}\right)\end{aligned}$$

Note that, if $\mathbf{h}$ is normalized, *i.e.* $\sum_{i=1}^{i=N} \mathbf{h}[i] = 1$, $\mathbf{H}^\top\mathbf{1}$ is an $N \times 1$ all one vector. The equation above can be simplified as:

$$\mathbf{x}^{k+1} = \mathbf{x}^k \times \mathbf{H}^\top\left(\frac{\mathbf{y}}{\mathbf{H}\mathbf{x}^k}\right),$$

To deal with the noisy measurement with Gaussian noise, iterative algorithms such as Landweber algorithm [5] and fast iterative soft-thresholding (FISTA) can be applied.

As we can see in the algorithm above, the deconvolution is performed through implementing two convolution operations in each iteration. However, the convolution is computationally expensive partially because the convolution kernel $\mathbf{h}$ and the image $\mathbf{x}$ have the same size and are big. To decrease the cost of deconvolution, it is essential to find a more efficient way to perform convolution.

2.3 Efficient Convolution

2.3.1 Convolution algorithms for 1D signal

2.3.1.1 Direct convolution

For direct 1D convolution with signal $\mathbf{x}$ with size N and kernel $\mathbf{h}$ of size K, the operation is defined as:

$$(\mathbf{x} * \mathbf{h})[n] = \sum_{m=0}^{K-1} \mathbf{x}[m-m]\mathbf{h}[m],$$

where each output sample requires K multiplications and additions, the computational complexity is $O(NK)$. If not considering the intermediate storage, direct convolution requires $C(N+K)$ of memory storage. Here C is the memory taken for each pixel. It is a constant that depends on the required precision (e.g., single, double precision).

2.3.1.2 FFT-based convolution

FFT-based convolution is an efficient method for performing convolution by leveraging the properties of the Fourier transform. According to the convolution theorem, convolution in the spatial domain can be transformed into element-wise multiplication in the frequency domain. Given a signal $\mathbf{x}$ and a filter $\mathbf{h}$, the convolution can be computed as

$$\mathbf{y} = \mathbf{F}^{-1}\left(\mathbf{F}\mathbf{x} \times \mathbf{F}\mathbf{h}\right).$$

This approach significantly reduces the computational complexity, especially for large datasets, compared to direct convolution in the spatial domain.

The steps for FFT-based convolution are as follows:

1. $\mathbf{x}$ and $\mathbf{h}$ are first padded to size $N_p = N + K - 1$. At this step, it requires memory to store $\mathbf{x}$, $\mathbf{h}$, and 2 zero-padded variable with size N_p. In total, it requires $C(N + K + 2N_p)$ of memory.

2. FFT is applied to the 2 zero-padded data, with a complexity of $O(N_p \log N_p)$. Each N_p-point FFT result requires $2CN_p$ of storage for complex value. Combining the memory required for storing variables from the earlier step, the total memory required at this step is:

$$C(N + K + 2N_p) + 2CN_p \times 2 = C(N + K + 6N_p).$$

3. Element-wise multiplication is performed in the frequency domain, with a complexity of $O(N_p)$. The previous storage from step 1 for the two zero-padded results, $2CN_p$, can be released, and the result of the complex multiplication requires an additional $2CN_p$ of storage. Thus, the total memory requirement at this step is

$$C(N + K + 6N_p) + 2CN_p - 2CN_p = C(N + K + 6N_p).$$

4. Apply IFFT to the multiplication result, with a complexity of $O(N_p \log N_p)$. It requires an additional CN_p of memory to store the IFFT results, and the previous storage of $4CN_p$ for the two FFT results from the previous step can be released. Therefore, the memory requirement at this step becomes

$$C(N + K + 6N_p) - 4CN_p + CN = C(2N + K + 2N_p).$$

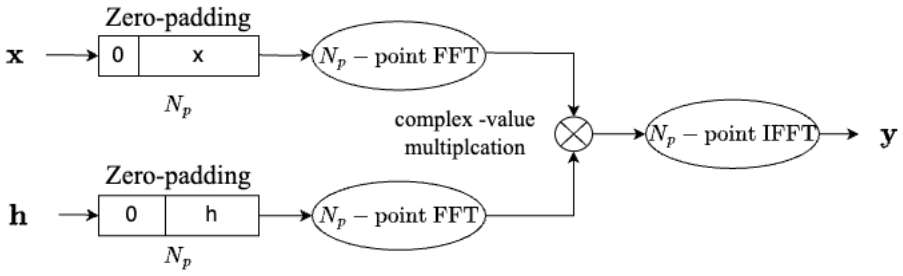


Figure 2.2: FFT-based convolution

This process is illustrated in Figure 2.2, and the memory estimation is shown in Figure 2.3. Combining these steps, the total complexity of FFT-based convolution is:

$$O(N_p \log N_p) + O(N_p \log N_p) + O(N_p) + O(N_p \log N_p) \approx O(N_p \log N_p).$$

Peak memory refers to the maximum amount of memory an algorithm uses at any point during its execution. Understanding peak memory usage helps identify potential memory bottlenecks and prevent crashes due to out-of-memory errors.

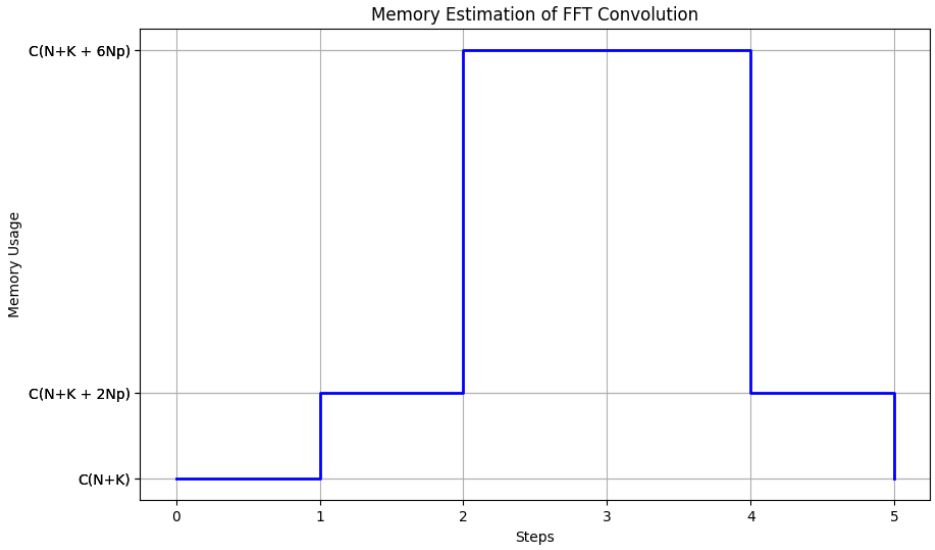


Figure 2.3: Memory estimation of FFT-based convolution

The theoretical peak memory usage is given by:

$$C(N + K + 6N_p) \approx 7CN_p.$$

2.3.1.3 Overlap-add convolution

The overlap-add method is a technique used to efficiently perform convolution on large signals by breaking the input signal into smaller segments. Each segment is convolved with the kernel, and the resulting outputs are combined by overlapping and adding the partial results. This method reduces computational complexity for long signals. By processing the signal in blocks, overlap-add is particularly useful in applications where memory and speed are critical, such as real-time signal processing. The detailed steps for overlap and add convolution is shown in Figure 2.4 and its memory estimation is shown in Figure 2.5. The detailed steps are as follows:

1. The input signal $\mathbf{x}$ is segmented into smaller blocks of length L. L is chosen such that $N_p \geq L + K - 1$, where N_p is the next higher number composed of small prime factors $2, 3, 5, 7$ for optimally sized FFT transform [6].

   The initial memory storage required is $C(N + K)$ to store $\mathbf{x}$ and $\mathbf{h}$.

2. Both the signal block and the kernel are zero-padded to length N_{opt}. N_{opt} is defined as:

   $N_{\text{opt}} = \min\{N_p \mid N_p \geq L+K-1$ and N_p is the next higher number with prime factors $2, 3, 5, 7\}$.

   The memory required to store the zero-padded results for both the signal block and the kernel is $2CN_{\text{opt}}$. Therefore, the memory usage at this step is:

   $$C(N + K) + 2CN_{\text{opt}} = C(N + K + 2N_{\text{opt}}).$$

3. Compute the N_{opt}-point FFT on the two zero-padded blocks. The computational complexity of performing the FFT on the two zero-padded blocks is $O(2N_{\text{opt}} \log N_{\text{opt}})$. This step requires $4CN_{\text{opt}}$ to store the results of the FFT of the two blocks. Thus, the memory usage at this step is:

$$C(N + K + 2N_{\text{opt}}) + 4CN_{\text{opt}} = C(N + K + 6N_{\text{opt}}).$$

4. Perform complex-valued multiplication in the frequency domain. The computational complexity of this operation is $O(N_{\text{opt}})$. It requires an additional $2CN_{\text{opt}}$ of memory to store the multiplication results. Since the memory for the zero-padded blocks from step 2 can be released, the memory usage at this step remains:

$$C(N + K + 6N_{\text{opt}}) + 2CN_{\text{opt}} - 2CN_{\text{opt}} = C(N + K + 6N_{\text{opt}}).$$

5. Perform an IFFT on the multiplication result. The computational complexity is $O(N_{\text{opt}} \log N_{\text{opt}})$. The IFFT result requires CN_{opt} of storage, and the memory used for storing the FFT results from step 3 can be released. Therefore, the memory usage at this step becomes:

$$C(N + K + 6N_{\text{opt}}) + CN_{\text{opt}} - 4CN_{\text{opt}} = C(N + K + 3N_{\text{opt}}).$$

6. Perform overlap-add with the previous convolution block. The computational complexity is $O(N_{\text{opt}})$. It requires CN of memory to store the overlap-add results, and the memory for storing the multiplication results for step 3 can be released. The memory usage at this step is:

$$C(N + K + 3N_{\text{opt}}) + CN - 2CN_{\text{opt}} = C(2N + K + N_{\text{opt}}).$$

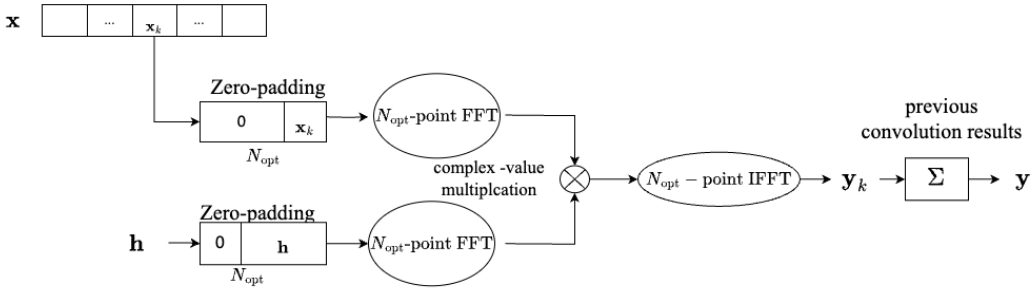


Figure 2.4: Overlap add convolution

Repeat the convolutions with segment blocks $\lceil \frac{N}{L} \rceil$ times. Combining the steps above and assuming that $N_{\text{opt}} \approx L \approx K$, the computational complexity of overlap and add convolution is:

$$O(\lceil \frac{N}{L} \rceil N_{\text{opt}} \log(N_{\text{opt}}) + \lceil \frac{N}{L} \rceil N_{\text{opt}}) \approx O(N(\log(N_{\text{opt}}) + 1)) \approx O(N \log(K)).$$

And the peak memory is: $C(2N + K + N_{\text{opt}}) \approx 2C(N + K)$

Comparing the three convolution algorithms, it is more efficient to use overlap and add convolution method when $K \ll N$.

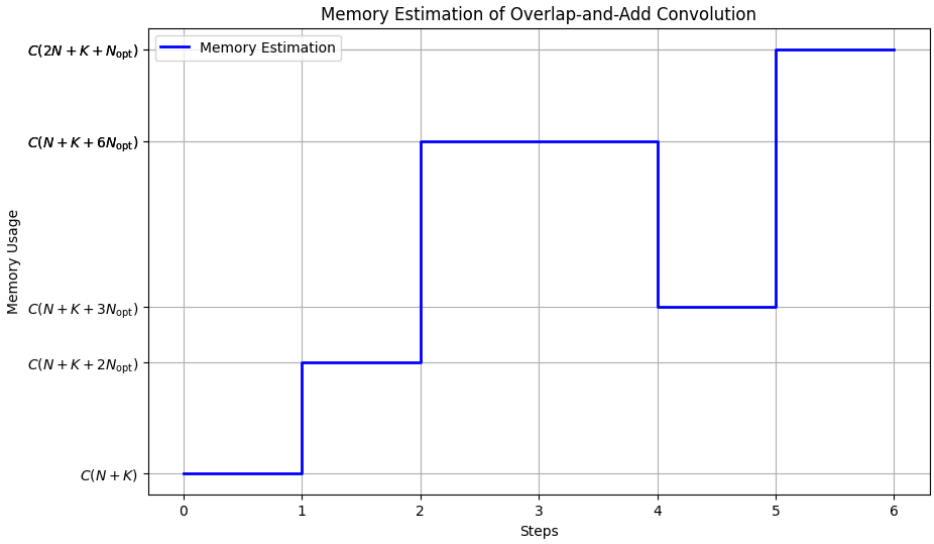


Figure 2.5: Memory estimation of overlap add convolution

2.3.2 Convolution methods with d-dimensional signal

Similar to the 1D case, the convolution methods discussed in Section 2.3.1 can be extended to d-dimensional signals [7]. Consider d-dimensional signals $x, y, h \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, with $N = \prod_{i=1}^{d} n_i$. Let $[i_1, i_2, \ldots, i_d]$ represent the indices of a d-dimensional array. The formula for d-dimensional convolution between a d-dimensional signal x and a d-dimensional kernel h is given by:

$$y[i_1, i_2, \ldots, i_d] = \sum_{j_1=1}^{n_1} \sum_{j_2=1}^{n_2} \cdots \sum_{j_d=1}^{n_d} x[i_1 - j_1, i_2 - j_2, \ldots, i_d - j_d] \cdot h[j_1, j_2, \ldots, j_d]$$

2.3.2.1 FFT-based convolution for d-dimensional signals

The Discrete Fourier Transform of a d-dimensional signal $x \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is given by [8]:

$$X[l_1, l_2, \ldots, l_d] = \sum_{i_1=0}^{n_1-1} \sum_{i_2=0}^{n_2-1} \cdots \sum_{i_d=0}^{n_d-1} x[i_1, i_2, \ldots, i_d] \cdot \exp\left(-2\pi j \left(\frac{l_1 i_1}{n_1} + \frac{l_2 i_2}{n_2} + \cdots + \frac{l_d i_d}{n_d}\right)\right)$$

The inverse DFT is given by:

$$x[i_1, i_2, \ldots, i_d] = \frac{1}{N} \sum_{l_1=0}^{n_1-1} \sum_{l_2=0}^{n_2-1} \cdots \sum_{l_d=0}^{n_d-1} X[l_1, l_2, \ldots, l_d] \cdot \exp\left(2\pi j \left(\frac{l_1 i_1}{n_1} + \frac{l_2 i_2}{n_2} + \cdots + \frac{l_d i_d}{n_d}\right)\right).$$

The convolution of two d-dimensional signals x and h is defined as:

$$y[i_1, i_2, \ldots, i_d] = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \cdots \sum_{j_d=0}^{n_d-1} x[i_1 - j_1, i_2 - j_2, \ldots, i_d - j_d] \cdot h[j_1, j_2, \ldots, j_d]$$

We now show that convolution theorem for d-dimensional case. Denote $X[l_1, \ldots, l_d]$, $H[l_1, \ldots, l_d]$, and $Y[l_1, \ldots, l_d]$ the DFTs of $x[i_1, \ldots, i_d]$, $h[i_1, \ldots, i_d]$, and $y[i_1, \ldots, i_d]$, respectively. By applying the DFT to $y[i_1, i_2, \ldots, i_d]$, the convolution theorem for d-dimensional signal is derived as follows:

$$\begin{aligned}
Y[l_1, \ldots, l_d] &= \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_d=0}^{n_d-1} \left(\sum_{j_1=0}^{n_1-1} \cdots \sum_{j_d=0}^{n_d-1} \cdot [i_1 - j_1, \ldots, i_d - j_d] \cdot h[j_1, \ldots, j_d] \right) \\
&\quad \times \exp\left(2\pi j \left(\frac{l_1 i_1}{n_1} + \cdots + \frac{l_d i_d}{n_d}\right)\right) \\
&= \sum_{j_1=0}^{n_1-1} \cdots \sum_{j_d=0}^{n_d-1} h[j_1, \ldots, j_d] \sum_{i_1=0}^{n_1-1} \cdots \sum_{i_d=0}^{n_d-1} x[i_1 - j_1, \ldots, i_d - j_d] \\
&\quad \cdot \exp\left(-2\pi j \left(\frac{l_1 i_1}{n_1} + \cdots + \frac{l_d i_d}{n_d}\right)\right) \\
&= \sum_{j_1=0}^{n_1-1} \cdots \sum_{j_d=0}^{n_d-1} h[j_1, \ldots, j_d] \cdot X[l_1, l_2, \ldots, l_d] \cdot \exp\left(-2\pi j \left(\frac{l_1 j_1}{n_1} + \cdots + \frac{l_d j_d}{n_d}\right)\right) \\
&\qquad \text{(Shift Theorem)} \\
&= X[l_1, \ldots, l_d] \sum_{j_1=0}^{n_1-1} \cdots \sum_{j_d=0}^{n_d-1} h[j_1, \ldots, j_d] \cdot \exp\left(-2\pi j \left(\frac{l_1 j_1}{n_1} + \cdots + \frac{l_d j_d}{n_d}\right)\right) \\
&= X[l_1, \ldots, l_d] H[l_1, \ldots, l_d]
\end{aligned}$$

Based on the d-dimensional convolution theorem, FFT-based convolution can be extended to d-dimensions. The steps for convolving a d-dimensional input signal $x[i_1, \ldots, i_d] \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with a kernel $h[i_1, \ldots, i_d] \in \mathbb{R}^{k_1 \times \cdots \times k_d}$ are as follows:

1. Zero padded the $x[i_1, \cdots, i_d]$ and $h[i_1, \cdots, i_d]$ to size $(n_1 + k_1 - 1) \times \cdots (n_d + k_d - 1)$.
2. Perform d-dimensional FFT on the zero-padded signals $x[i_1, \cdots, i_d]$ and $h[i_1, \cdots, i_d]$ to obtain $X[l_1, \cdots, l_d]$ and $H[l_1, \cdots, l_d]$, respectively.
3. Multiply the Fourier transforms element-wise:
$$Y[l_1, l_2, \cdots, l_d] = X[l_1, l_2, \cdots, l_d] \cdot H[l_1, l_2, \cdots, l_d]$$
4. Perform the inverse d-dimensional FFT on $Y[l_1, \cdots, l_d]$ to obtain $y[i_1, \cdots, i_d]$, and trim the result to size $n_1 \times n_2 \times \cdots \times n_d$, discarding the padded regions.

We define $N_{p_i} = n_i + k_i - 1$ as the size of zero-padding in i^{th} dimension, and let $N_p = \prod_{i=1}^{d} n_i + k_i - 1$, then the computational complexity for d-dimensional convolution is $O(\prod_{i=1}^{d} N_{p_i} \log(\prod_{i=1}^{d} N_{p_i})) = O(N_p \log N_p)$. And theoretical peak memory is $7CN_p$.

2.3.2.2 Overlap-add convolution for d-dimensional data

Similarly, the overlap-add method can also be extended to d-dimensional signals. The steps are as follows:

1. The input signal $x[i_1, i_2, \ldots, i_d]$ is segmented into smaller blocks of size $L_1 \times L_2 \times \cdots \times L_d$. Each L_i is chosen such that $N_{p_i} \geq L_i + K_i - 1$ is the next higher number comprised of small prime factors $2, 3, 5, 7$ for each dimension.
2. Both the signal block and the kernel are zero-padded to size $N_{\text{opt}1} \times N_{\text{opt}2} \times \cdots \times N_{\text{opt}d}$, where $N_{\text{opt}i}$ is defined as:

   $N_{\text{opt}i} = \min\{N_{p_i} \mid N_{p_i} \geq L_i + K_i - 1$ and N_{p_i} is the next higher number with prime factors $2, 3, 5$
3. Compute the d-dimensional $N_{\text{opt}1} \times N_{\text{opt}2} \times \cdots \times N_{\text{opt}d}$-point FFT on the two zero-padded blocks (input and kernel).
4. Perform complex-valued multiplication in the frequency domain.
5. Perform inverse FFT (IFFT) on the multiplication result to bring it back to the spatial domain.
6. Perform the overlap-add operation with the previous convolution blocks.

2.3.2.3 Separable convolution

For d-dimensional convolution, the convolution cost can be further reduced if the multidimensional filter can be decomposed into multiple lower-dimensional filters. This reduces the computational cost significantly, as it allows the d-dimensional convolution to be performed as a series of 1D convolutions along each dimension.

In separable convolution, the d-dimensional kernel h is decomposed into d 1D kernels $h_1, h_2, \ldots, h_d$. h is separable if:

$$h[j_1, j_2, \ldots, j_d] = h_1[j_1] \cdot h_2[j_2] \cdot \cdots \cdot h_d[j_d]$$

The separable convolution is computed as:

Using the separability of h, the convolution of x with h becomes:

$$\begin{aligned} y[i_1, i_2, \ldots, i_d] &= \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \cdots \sum_{j_d=0}^{n_d-1} x[i_1 - j_1, i_2 - j_2, \ldots, i_d - j_d] \cdot h_1[j_1] \cdot h_2[j_2] \cdot \cdots \cdot h_d[j_d] \\ &= (((x * h_1) * h_2) * \cdots * h_d)\,[i_1, i_2, \ldots, i_d] \end{aligned}$$

This means that instead of performing a full d-dimensional convolution, the operation performs d times 1D convolutions along each dimension. It has computational complexity of $O(\sum_{i=1}^{d} n_i \cdot k_i)$ for direct convolution, $O(\sum_{i=1}^{d} N_{p_i} \log N_{p_i})$ for FFT-based convolution, and $O(\sum_{i=1}^{d} n_i \log k_i)$ for overlap-add convolution.

References

[1] G. M. van Kempen and L. J. van Vliet, "Background estimation in nonlinear image restoration," *JOSA A*, vol. 17, no. 3, pp. 425–433, 2000.

[2] N. Dey, L. Blanc-Feraud, C. Zimmer, *et al.*, "Richardson-Lucy algorithm with total variation regularization for 3d confocal microscope deconvolution," *Microscopy research and technique*, vol. 69, no. 4, pp. 260–266, 2006.

[3] W. H. Richardson, "Bayesian-based iterative method of image restoration," *JoSA*, vol. 62, no. 1, pp. 55–59, 1972.

[4] L. B. Lucy, "An iterative technique for the rectification of observed distributions," *Astronomical Journal, Vol. 79, p. 745 (1974)*, vol. 79, p. 745, 1974.

[5] L. Landweber, "An iteration formula for fredholm integral equations of the first kind," *American Journal of Mathematics*, vol. 73, no. 3, pp. 615–624, 1951, ISSN: 00029327, 10806377. [Online]. Available: http://www.jstor.org/stable/2372313.

[6] F. Wefers, *Partitioned convolution algorithms for real-time auralization*. Logos Verlag Berlin GmbH, 2015, vol. 20.

[7] D. E. Dudgeon, "Multidimensional digital signal processing," *Engewood Cliffs*, 1983.

[8] R. Bracewell, "The two-dimensional convolution theorem," in *Fourier Analysis and Imaging*. Boston, MA: Springer US, 2003, pp. 204–221, ISBN: 978-1-4419-8963-5. DOI: 10.1007/978-1-4419-8963-5_6. [Online]. Available: https://doi.org/10.1007/978-1-4419-8963-5_6.

3 Method

In this chapter, we propose a multiresolution convolution scheme. Our method utilizes the special structure of the PSF and applies the multi-rate signal processing methods [1]. Briefly, it retains the center part of the PSF, which contains the majority of the PSF energy and downsamples the sidelobes. Section 3.2 provides a detailed explanation of the algorithm. For simplicity, the models are explained first in 1D, then generalized to the d-dimensional case. In Section 3.3, the theoretical computational cost of the algorithm is estimated and compared with other schemes.

3.1 Presentation of the main idea

The proposed convolution method leverages the special structure of the PSF in conventional microscopy modalities (e.g., confocal, wide field, etc.). As seen in Figure 3.1, the central part of the PSF, denoted as $\mathbf{h}_c$, is relatively narrow and contains the largest part of the energy, while the two sidelobes, denoted as $\mathbf{h}_s$, spread out wide and contain less energy. In general, the PSF can be decomposed as $\mathbf{h} = \mathbf{h}_c + \mathbf{h}_s$.

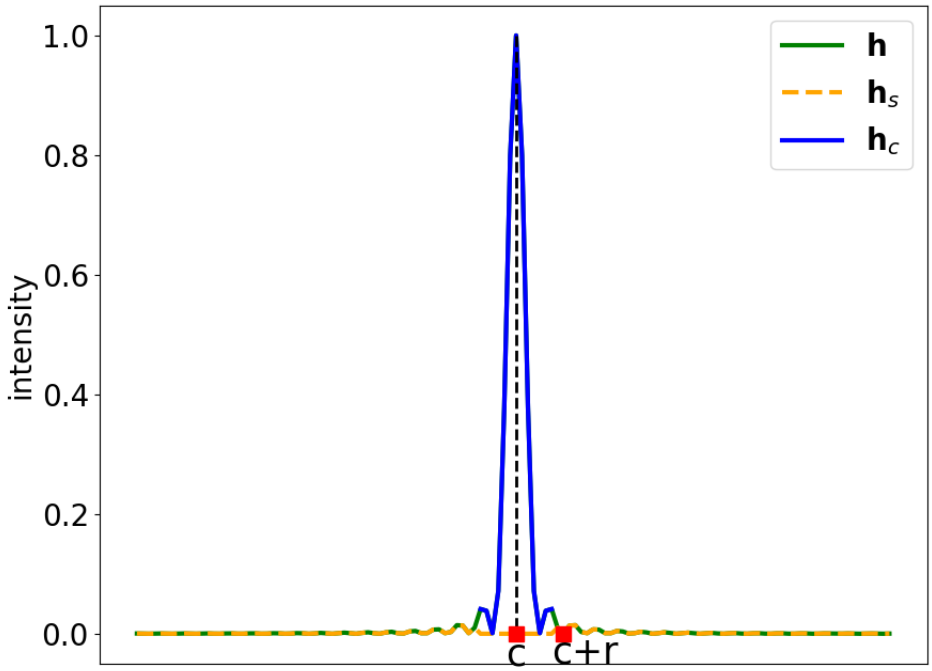


Figure 3.1: 1D PSF

The reference forward model, using the full PSF as the convolution kernel, is defined as $\mathbf{y}_{\text{ref}}$. It can be written as:

$$\mathbf{y}_{\text{ref}} = \mathbf{x} * (\mathbf{h}_c + \mathbf{h}_s) = \mathbf{x} * \mathbf{h}_c + \mathbf{x} * \mathbf{h}_s,$$

In linear operator form, this becomes:

$$\mathbf{y}_{\text{ref}} = \mathbf{H}_c\mathbf{x} + \mathbf{H}_s\mathbf{x}.$$

Here, $\mathbf{H}_c$ and $\mathbf{H}_s$ are the linear operators that model convolution with $\mathbf{h}_c$ and $\mathbf{h}_s$, respectively.

Since convolution is a linear operation, we can separate the PSF and perform the convolution independently. This separation involves two convolutions: one with a much smaller kernel $\mathbf{h}_c$ and another with the original kernel $\mathbf{h}_s$. The kernel $\mathbf{h}_c$, with size $K_c = 2r + 1 \ll N$, contains most of the PSF's energy. In practice, due to a lack of user-friendly efficient tools, scientists are limited to using only the central part as the convolution kernel, to be able to have a memory-wise feasible convolution with reasonable deconvolution time. In other words, these forward models approximate $\mathbf{y}_{\text{ref}}$ using $\mathbf{H}_c\mathbf{x}$, denoted as $\mathbf{y}_{\text{c}}$:

$$\mathbf{y}_{\text{c}} = \mathbf{H}_c\mathbf{x}$$

3.2 Multi-resolution convolution scheme

The convolution scheme $\mathbf{y}_{\text{c}}$ discards the two sidelobes of the PSF, which impacts the accuracy of the convolution. To achieve high precision while reducing computational costs, we implemented decimation methods [1] on $\mathbf{h}_s$ and $\mathbf{x}$, then performed convolution with the decimated data. The detailed process is shown in Figure 3.2.

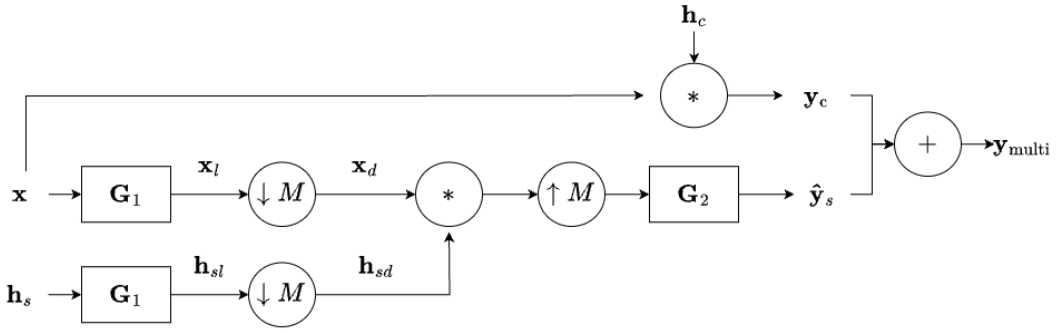


Figure 3.2: Flow diagram of multi-resolution convolution method

3.2.1 Steps of the Multi-resolution convolution scheme

1. **Decompose PSF**

First, separate the PSF $\mathbf{h}$ into components $\mathbf{h}_c$ and $\mathbf{h}_s$. To avoid artifacts caused by edges, a window is applied to smooth the edges. Here, Tukey window, also known as the tapered cosine window, is chosen because of its flexibility in adjusting the shape of the window. For 1D PSF, we assume that $\mathbf{h}$ reaches the maximum intensity at location c, and r is a radius parameter. The Tukey window with length $K_c = 2r + 1$, denoted as $\mathbf{w}$, is defined as [2]:

$$\mathbf{w}[n] = \begin{cases} \frac{1}{2}\left[1 + \cos\left(\pi\left(\frac{2n}{\alpha(K_c-1)} - 1\right)\right)\right] & 0 \le n < \frac{\alpha(K_c-1)}{2} \\ 1 & \frac{\alpha(K_c-1)}{2} \le n \le (K_c-1)\left(1-\frac{\alpha}{2}\right) \\ \frac{1}{2}\left[1 + \cos\left(\pi\left(\frac{2n}{\alpha(K_c-1)} - \frac{2}{\alpha} + 1\right)\right)\right] & (K_c-1)\left(1-\frac{\alpha}{2}\right) < n \le (K_c-1) \end{cases}$$

Here, α is the shape parameter of the window, ranging from 0 to 1. When $\alpha = 0$ $\mathbf{w}$ becomes a rectangular window, and when $\alpha = 1$ it is a Hanning window as shown in Figure 3.3.

Define window $\mathbf{w}_c \in R^N$ and $\mathbf{w}_s \in R^N$ as:

$$\mathbf{w}_c[k] = \begin{cases} \mathbf{w}[k] & \text{if } c - r \le k \le c + r, \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathbf{w}_s[k] = \begin{cases} 1 - \mathbf{w}[k] & \text{if } c - r \le k \le c + r, \\ 1 & \text{otherwise.} \end{cases}$$

Let $\times$ denote the element-wise multiplication. Then, $\mathbf{h}_c$ and $\mathbf{h}_s$ are calculated as follows:

$$\mathbf{h}_c = \mathbf{h} \times \mathbf{w}_c,$$
$$\mathbf{h}_s = \mathbf{h} \times \mathbf{w}_s.$$

2. **Decimation on x and $\mathbf{h}_s$**

   The computational cost of convolution with sidelobes $\mathbf{h}_s$ can be significantly reduced by downsampling. Since $\mathbf{h}_s$ contains a limited amount of energy, and $\mathbf{h}_s$ is typically band-limited lowpass, downsampling in this region will not cause a huge reduction in accuracy.

   Thus, the next step is to decimate $\mathbf{x}$ and $\mathbf{h}_s$ by a factor of M. To avoid aliasing artifacts caused by downsampling, a low-pass filter $\mathbf{g}_1$ with cutoff frequency $\omega_c = \frac{\pi}{M}$ should be applied before downsampling.

   Here, the low-pass filter $\mathbf{g}_1$ is implemented as a Finite Impulse Response (FIR) filter, and it is designed using the window method with Kaiser window. Define its transition width $\Delta\omega = 0.2\pi$ rad/sample, and stopband attenuation $A = 40$ dB. Then the filter length $N_{\mathbf{g}_1}$ is calculated by [1]:

$$N_{\mathbf{g}_1} = \lceil \frac{A - 8}{2.285\Delta\omega} + 1 \rceil = 23,$$

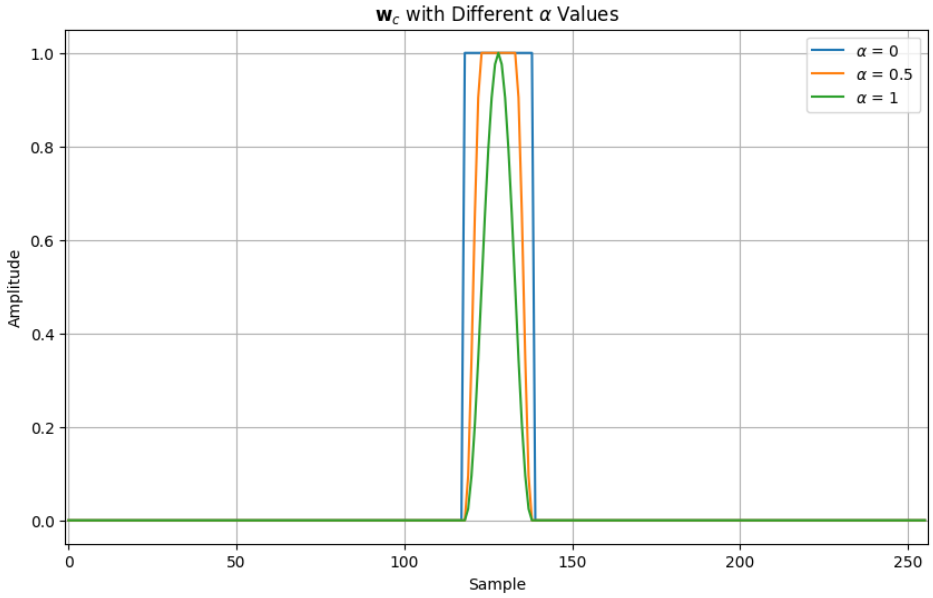


Figure 3.3: Window $\mathbf{w}_c$ with different α values

The Kaiser window's shape is controlled by the parameter β, which depends on the desired stopband attenuation A. The empirical formula for β is given by:

$$\beta = \begin{cases} 0.1102(A - 8.7), & \text{if } A > 50 \text{ dB} \\ 0.5842(A - 21)^{0.4} + 0.07886(A - 21), & \text{if } 21 \leq A \leq 50 \text{ dB} \\ 0, & \text{if } A < 21 \text{ dB} \end{cases}$$

For $A = 40$ dB,

$$\beta = 0.5842(A - 21)^{0.4} + 0.07886(A - 21) \approx 3.395.$$

Let $\alpha = \frac{N_{g_1}-1}{2}$, and $I_0(\beta)$ is the modified zeroth-order Bessel function of the first kind, $\mathbf{g}_1$ is obtained through:

$$\mathbf{g}_1[n] = \begin{cases} \frac{\sin(\omega_c(n-\alpha))}{\pi(n-\alpha)} \cdot \frac{I_0\left(\beta\sqrt{1-\left(\frac{n-\alpha}{\alpha}\right)^2}\right)}{I_0(\beta)}, & 0 \leq n \leq N_{g_1} - 1 \\ 0, & \text{otherwise.} \end{cases}$$

The frequency response of $\mathbf{g}_1$ with cutoff frequency $\omega_c = \frac{pi}{2}$ and the spectrum of an example $\mathbf{h}_s$ are shown in Figure 3.4. From the Figure, $\mathbf{h}_s$ is a band-limited signal, and the loss of $\mathbf{h}_s$ caused by applying low-pass filter is negligible.

The filtering process with $\mathbf{g}_1$ process can be written as a linear operator in matrix form $\mathbf{G}_1$.

Let $\mathbf{x}_l$ and $\mathbf{h}_{sl}$ denote the image data after applying low-pass filter.

$$\mathbf{x}_l = \mathbf{G}_1\mathbf{x}, \quad \mathbf{h}_{sl} = \mathbf{G}_1\mathbf{h}_s$$

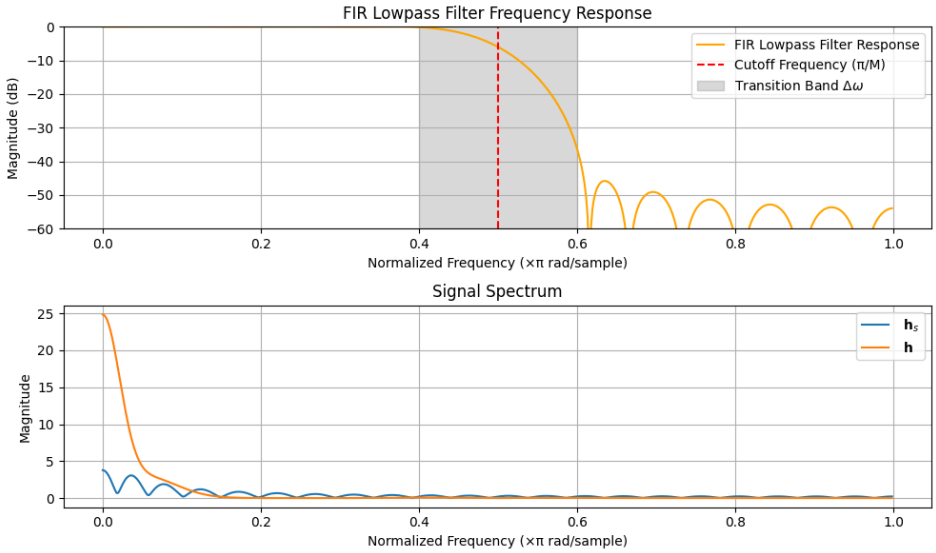


Figure 3.4: Frequency response of $\mathbf{g}_1$ and the spectrum of $\mathbf{h}_s$

$\mathbf{x}_l$ and $\mathbf{h}_{sl}$ are then downsampled by factor of M. The downsampling by M operator can be written as matrix form $\mathbf{D}_M$. Assuming that N is a multiplier of M, $\mathbf{D}_M$ is an $\frac{N}{M} \times N$ matrix, For example, when $M = 2$, $\mathbf{D}_2$ has this form:

$$\mathbf{D}_2 = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 1 & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & 0 & 1 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

Let $\mathbf{x}_d$ and $\mathbf{h}_{sd}$ denote the signals after downsampling. Both $\mathbf{x}_d$ and $\mathbf{h}_{sd}$ are of the size $\frac{N}{M}$ and can be written as:

$$\mathbf{x}_d = \mathbf{D}_M \mathbf{x}_l, \quad \mathbf{x}_d[n] = \mathbf{x}_l[Mn],$$

$$\mathbf{h}_{sd} = \mathbf{D}_M \mathbf{h}_{sl}, \quad \mathbf{h}_{sd}[n] = \mathbf{h}_{sl}[Mn].$$

3. **Convolve the decimated data**

   Next, the decimated data $\mathbf{x}_d$ is convolved with $\mathbf{h}_{sd}$, producing output $\mathbf{y}_{sd}$. We define the $\frac{N}{M} \times \frac{N}{M}$ matrix $\mathbf{H}_{sd}$, which models the convolution with $\mathbf{h}_{sd}$. This process is expressed by:

$$\mathbf{y}_{sd} = \mathbf{H}_{sd} \mathbf{x}_d.$$

4. **Upsample the convolution output data**

   Next, $\mathbf{y}_{sd}$ is upsampled by a factor of M. The upsampling by M operation can be written as a $N \times \frac{N}{M}$ matrix $\mathbf{U}_M$. $\mathbf{U}_M$ is the adjoint of $\mathbf{D}_M$, $\mathbf{U}_M = \mathbf{D}_M^T$.

Then an interpolation filter $\mathbf{g}_2$ is implemented to smooth out the discontinuities caused by the inserted zeros [3]. $\mathbf{g}_2$ is a low-pass filter with cutoff frequency $\frac{\pi}{M}$. Here, $\mathbf{g}_2$ is implemented as a FIR Kaiser window. It has order $2a * M$, where a is the number of the neighbor terms that will be taken into the interpolation. If we take $a = 2$, the length of $\mathbf{g}_2$ is $N_{g_2} = 4M + 1$. The linear operator for convolution with the filter $\mathbf{g}_2$ is defined as $\mathbf{G}_2$. We define the interpolation output as $\hat{\mathbf{y}}_s$, and it is calculated as:

$$\hat{\mathbf{y}}_s = \mathbf{G}_2 \mathbf{U}_M \mathbf{y}_{sd}$$

In conclusion, combining the steps described above, the multi-resolution convolution scheme is written as:

$$\mathbf{y}_{\text{multi}} = \mathbf{y}_c + \hat{\mathbf{y}}_s = \mathbf{H}_c \mathbf{x} + \mathbf{G}_2 \mathbf{U}_M (\mathbf{H}_{sd} \mathbf{x}_d) = \mathbf{H}_c \mathbf{x} + \hat{\mathbf{H}}_s \mathbf{x}.$$

And we can see it schematically in Figure 3.2.

3.2.2 Further reduce the computational cost through the polyphase decomposition

To further reduce the computation cost, in step 2 above, instead of first lowpass filtering and then downsampling on $\mathbf{x}$ and $\mathbf{h}_s$, we implement the polyphase representation of decimation[4]. This process is illustrated in Figure 3.5.

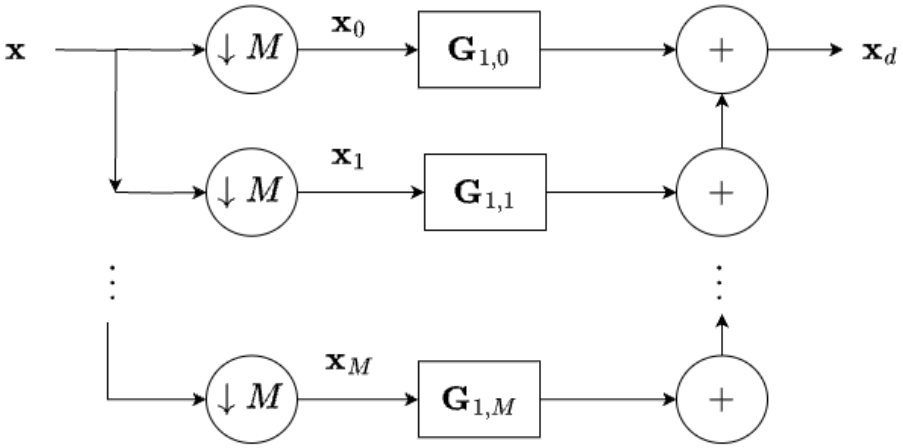


Figure 3.5: Polyphase representation of decimation on $\mathbf{x}$

Note that in theory, the computation cost can be further reduced through the polyphase representation of the interpolation step. However, it is not employed in this project, because it increased the error.

Denote the low-pass-filtered and downsampled sequence of $\mathbf{x}$ as $\mathbf{x}_d$. We perform polyphase decomposition on $\mathbf{g}_1[n]$, and divide it into M sub-filters, $\mathbf{g}_{1,m}$, for $m = 0, 1, \ldots, M-1$. For each m, the polyphase component $\mathbf{g}_{i,m}$ of $\mathbf{g}_1$ is obtained through

first left shifting $\mathbf{g}_1$ by m, then downsampling by M.

$$\mathbf{g}_{1,m}[n] = \mathbf{g}_1[nM + m]$$

Similarly, the polyphase component $\mathbf{x}_m$ of $\mathbf{x}$ is obtained through right shifting $\mathbf{x}$ by m then downsampling by M.

$$\mathbf{x}_m[n] = \mathbf{x}[nM - m]$$

Then the decimation output $\mathbf{x}_d$ can be written as:

$$\begin{aligned}
\mathbf{x}_d[n] &= \sum_{i=1}^{N_{g_1}-1} \mathbf{g}_1[i]\mathbf{x}[nM - i] \\
&= \sum_{m=0}^{M-1} \sum_{j=0}^{\lceil \frac{N_{\mathbf{g}_1}}{M} \rceil - 1} \mathbf{g}_1[jM + m]\mathbf{x}[nM - (jM + m)] \\
&= \sum_{m=0}^{M-1} \left(\sum_{j=0}^{\lceil \frac{N_{\mathbf{g}_1}}{M} \rceil - 1} \mathbf{g}_{1,m}[j]\mathbf{x}_m[n - j] \right)
\end{aligned}$$

Define the shifted and downsampled linear operator as $\mathbf{D}_{M,m}$, $m = 0, 1, \cdots M-1$. For $M = 2$, $\mathbf{D}_{2,0}$ and $\mathbf{D}_{2,1}$ have the following form:

$$\mathbf{D}_{2,0} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix},$$

$$\mathbf{D}_{2,1} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}.$$

Define the operator for convolution with polyphase components of $\mathbf{g}_1$ as $\mathbf{G}_{1,m}$, $m = 0, 1, \cdots M-1$. Then $\mathbf{x}_d$ is calculated as:

$$\mathbf{x}_d = \sum_{m=0}^{M-1} \mathbf{G}_{1,m}\mathbf{D}_{M,m}\mathbf{x}$$

Similarly, perform polyphase decimination on $\mathbf{h}_s$, and the decimination out put $\mathbf{h}_{sd}$ is calculated as:

$$\mathbf{h}_{sd} = \sum_{m=0}^{M-1} \mathbf{G}_{1,m}\mathbf{D}_{M,m}\mathbf{h}_s$$

Instead of filtering at the original high sampling rate followed by downsampling, polyphase decomposition performs M convolutions with downsampled data and subfilters. For the 1D case, consider the convolution between $\mathbf{x}$ and the filter $\gg_1$. Polyphase

decomposition reduces the cost of direct convolution from $O(N \cdot K_{g_1})$ to $O(N \cdot \frac{K_{g_1}}{M})$, where M is the decimation factor. In FFT-based convolution, it reduces the input size to smaller FFTs of length N/M, lowering the complexity from $O(N \log N)$ to $O(N \log(N/M))$. Similarly, in overlap-and-add convolution, polyphase decomposition reduces segment sizes, thereby lowering the computation cost from $O(N \log K_{g_1})$ to $O(N \log \frac{K_{g_1}}{M})$. In all cases, computation is accelerated by breaking the original problem into multiple smaller convolutions on downsampled signals.

3.2.3 Generalization to d-Dimensional Data

The models above can be generalized to d-dimensional data, where $x, y \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ with $\prod_{i=1}^{d} n_i = N$, and $h \in \mathbb{R}^{k_1 \times \cdots \times k_d}$ with $\prod_{i=1}^{d} k_i = K$. The vectorized data is denoted as $\mathbf{x}, \mathbf{y}, \mathbf{h}$.

The linear operators for 1D data can be generalized into vectorized d-dimensional data when the operation is separable across dimensions[5]. In this project, the linear operators for upsampling, downsampling, and separable convolution with lowpass filter $\mathbf{g}_1$ and $\mathbf{g}_2$ can be generalized to d-dimensional data. For vectorized d-dimensional data, the extended operator can be constructed using the Kronecker product of the original 1D operator.

Consider linear operators $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d$ that act on 1D vector $\mathbf{z}_1 \in \mathbb{R}_1^n, \mathbf{z}_2 \in \mathbb{R}_1^n, \ldots, \mathbf{z}_d \in \mathbb{R}_d^n$ respectively. Denote the Kronecker product as $\otimes$. The linear operator $\mathbf{A}$ that act on the vectorized d-dimensional data $\mathbf{z} \in \mathbb{R}^{n_1 \times n_2 \ldots n_d}$ can be written as:

$$\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_d \quad .$$

However, this generalization rule does not apply to convolutions with $\mathbf{h}$, $\mathbf{h}_c$, $\mathbf{h}_s$, and $\mathbf{h}_{sd}$. These convolutions are not separatable. For 1D data, $\mathbf{H}$, $\mathbf{H}_c$, $\mathbf{H}_s$ and $\mathbf{H}_{sd}$ are circular matrix. For vectorized d-dimensional data, these operators are constructed as block-circulant matrices with circulant blocks.

3.3 Computational cost of the three convolution schemes

Since we are working with large-sized data, we want to propose algorithms that can scale down the cost. In the following section, the computational complexity and the theoretical memory usage of the three convolution schemes are analyzed.

Consider vectorized d-dimensional image data $\mathbf{x}$ of size $N = \prod_{i=1}^{d} n_i$, vectorized d-dimensional convolution kernel $\mathbf{h}$ denote the of size K, $K = \prod_{i=1}^{d} k_i$. n_i is the size of $\mathbf{x}$ in the i^{th} dimension. k_i is the size of h in the i^{th} dimension. As discussed in section 2.3.1, for direct convolution, it has a computational complexity of $O(NK)$, and peak memory usage $C(N+K)$, where C represents memory per pixel. Define the size of zero padding, $N_p = \prod_{i=1}^{d}(n_i + k_i - 1)$. The FFT-based convolution achieves a more efficient complexity of $O(N_p \log N_p)$ but requires a theoretical peak memory of approximately

$7CN_p$. The overlap-add method has a complexity of $O(N \log K)$, and its peak memory usage is approximately $2C(N + K)$.

When comparing the three convolution algorithms, it is more efficient to use the overlap and add convolution method when $K \ll N$. When $K \approx N$ the overlap and add convolution are equivalent to FFT-convolution.

Now the costs for three convolution schemes, $\mathbf{y}_{\text{ref}}$, $\mathbf{y}_{\text{c}}$, and $\mathbf{y}_{\text{multi}}$ are calculated and compared.

3.3.1 Computational cost of $\mathbf{y}_{\text{ref}}$

For the convolution scheme using the full PSF as the convolution kernel, i.e. $\mathbf{y}_{\text{ref}}$, the kernel size K is equal to N, using overlap and add convolution will not improve efficiency. Thus, FFT-based convolution is implemented. The padding size is $N_p^{\text{ref}} = \prod_{i=1}^{d}(2n_i - 1) \approx 2^d \prod_{i=1}^{d} n_i = 2^d N$. The computational complexity for $\mathbf{y}_{\text{ref}}$ is:

$$O(N_p^{\text{ref}} \log(N_p^{\text{ref}})) \approx O(2^d N \log(2^d N)).$$

And the peak memory for $\mathbf{y}_{\text{ref}}$ is:

$$7CN_p^{\text{ref}} \approx 7 \cdot 2^d CN$$

3.3.2 Computational cost of $\mathbf{y}_{\text{c}}$

For the convolution scheme $\mathbf{y}_{\text{c}}$, which uses $\mathbf{h}_c$ as the convolution kernel. Define the size of $\mathbf{h}_c$ as K_c, and $K_c \ll N$, it is more efficient to use overlap and add convolution. The computational complexity is $O(N \log(K_c))$. The theoretical peak memory is $2C(N + K_c)$.

3.3.3 Computational cost of $\mathbf{y}_{\text{multi}}$

For the multi-resolution convolution scheme $\mathbf{y}_{\text{multi}}$, different types of convolution are implemented in multiple steps.

1. **The convolution of $\mathbf{x}$ and $\mathbf{h}_c$**: as discussed in 3.3.2, overlap and add convolution is implemented. Define K_c the size of $\mathbf{h}_c$. The computational complexity is: $O(N \log K_c)$ and the peak memory is $2C(N + K_c)$
2. **Convolution of polyphase components of lowpass filter $\mathbf{g}_1$ and $\mathbf{x}$**: the polyphase components of $\mathbf{g}_1$ and $\mathbf{x}$, denoted as as $\mathbf{g}_{1,i}$ and $\mathbf{x}_i$ respectively, where $i = 0, \cdots, M - 1,$. The size of $\mathbf{x}_i$ is $\prod_{i=1}^{d} \frac{n_i}{M} = \frac{N}{M^d}$, and the size of $\mathbf{g}_{1,i}$ is $\frac{N_{g_1}}{M}$.

   There are M convolutions for each polyphase component. The convolutions are separable and utilize the overlap-add method. The computational cost for convolving each polyphase component is given by:

$$O(\sum_{i=1}^{d} \frac{n_i}{M} \log(\frac{N_{g_1}}{M})).$$

In total, the M convolutions across polyphase components have computational complexity of:

$$M \cdot O(\sum_{i=1}^{d} \frac{n_i}{M} \log(\frac{N_{g_1}}{M})) = O(\sum_{i=1}^{d} n_i \log(\frac{N_{g_1}}{M})).$$

The peak memory required along dimension i is:

$$2C(n_i + \frac{N_{g_1}}{M})$$

Since we are applying separable convolution, the peak memory usage will be driven by the largest memory requirement across all dimensions. Typically, the dimension with the largest size n_i will dominate, so the peak memory usage is:

$$2C(\max(n_1, \cdots, n_d) + \frac{N_g}{M})$$

3. **Convolution of polyphase components of lowpass filter $\mathbf{g}_1$ and $\mathbf{h}_s$**: since $\mathbf{h}_s$ and $\mathbf{x}$ are of the same size, the computational cost is the same as that of the previous step.

4. Convolution of $\mathbf{x}_d$ and $\mathbf{h}_{sd}$: FFT-convolution is implemented. Both $\mathbf{x}_d$ and $\mathbf{h}_{sd}$ are of the size $\prod_{i=1}^{d} \frac{n_i}{M} = \frac{N}{M^d}$. The zero padding size for FFT-convolution is

$$N_p^{\text{multi}} = \prod_{i=1}^{d} (\frac{n_i}{M} + \frac{n_i}{M} - 1) \approx (\frac{2}{M})^d N.$$

The computational complexity is:

$$O(N_p^{\text{multi}} \log N_p^{\text{multi}}) = O((\frac{2}{M})^d N \log((\frac{2}{M})^d N)) \approx O((\frac{2}{M})^d N \log N).$$

The peak memory is $7CN_p = 7CN(\frac{2}{M})^d$.

5. Convolution of low-pass filter $\mathbf{g}_2$ and $\mathbf{y}_{sd}$: separable overlap and add convolution is implemented. The size of $\mathbf{y}_{sd}$ is $\prod_{i=1}^{d} n_i = N$ and the size of $\mathbf{g}_2$ is N_{g_2}. The computational complexity is: $\sum_{i=1}^{d} O(n_i \log N_{g_2})$, and the peak memory is $2C(\max(n1, \cdots, n_d) + N_{g_2})$.

Combining the steps above, $\mathbf{y}_{\text{multi}}$ has computational complexity:

$$O\left(N \log K_c + 2\sum_{i=1}^{d} n_i \log(\frac{N_{g_1}}{M}) + (\frac{2}{M})^d N \log N + \sum_{i=1}^{d} n_i \log N_{g_2}\right)$$
$$\approx O\left((\frac{2}{M})^d N \log N\right), \text{since } K_c, N_{\mathbf{g}_1}, N_{\mathbf{g}_2} \ll N.$$

The theoretical peak memory storage is computed by taking the maximum peak memory from each step:

$$\max\{2C(N+K_c), \frac{2C}{M^d}(N+N_{\mathbf{g}_1}), 7CN(\frac{2}{M})^d, 2C(N+(4M+1))\} = 7CN(\frac{2}{M})^d$$

Compared to the cost of $\mathbf{y}_{\text{ref}}$, $\mathbf{y}_{\text{multi}}$ decreases the computational complexity and peak memory by M^d.

In summary, the proposed multiresolution convolution scheme efficiently reduces computational cost by leveraging the special structure of the PSF. By retaining the central part of the PSF and downsampling the sidelobes, the method balances accuracy and computational efficiency. This approach is particularly useful for large-scale image data, where direct convolutions are computationally expensive. Typical decimation factors M range from 2 to 8, with higher values of M offering greater reductions in memory and computation, though at the cost of some accuracy. The use of polyphase decomposition further reduces computational cost by enabling smaller convolutions on downsampled data. This scheme significantly improves performance, particularly when combined with FFT-based and overlap-and-add convolutions.

References

[1] A. Oppenheim, "Discrete-time signal processing," *Prentice Hall google schola*, vol. 3, pp. 804–809, 1999.

[2] F. J. Harris, "On the use of windows for harmonic analysis with the discrete fourier transform," *Proceedings of the IEEE*, vol. 66, no. 1, pp. 51–83, 1978.

[3] F. J. Harris, *Multirate signal processing for communication systems*. River Publishers, 2022.

[4] B. Porat, *A Course in Digital Signal Processing*, 1st. USA: John Wiley & Sons, Inc., 1996, ISBN: 0471149616.

[5] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, vol. 51, no. 3, pp. 455–500, 2009.

4 Results

4.1 Performance metrics

In this project, we assess the computational cost and accuracy of each convolution algorithm. To assess computational cost, we use peak memory usage and computation time. To evaluate accuracy performances, three metrics are implemented: Normalized Mean Squared Error (NMSE), Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index (SSIM).

4.1.1 Normalized Mean Squared Error (NMSE)

Normalized Mean Squared Error (NMSE) is a metric used to quantify the difference between the original image and the reconstructed image. It is defined as:

$$\text{NMSE} = \frac{\sum_{i,j} \left(x_{ij} - \hat{x}_{ij}\right)^2}{\sum_{i,j} x_{ij}^2} \tag{4.1}$$

where x_{ij} represents the intensity value of the original image at pixel (i, j), and $\hat{x}_{ij}$ represents the intensity value of the reconstructed image at the same pixel. NMSE provides a measure of the approximaion error, normalized by the energy of the original image, making it useful for comparing the performance of different deconvolution methods regardless of the image's absolute intensity values.

4.1.2 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) is another metric used to measure the quality of the reconstructed image. PSNR is defined as:

$$\text{PSNR} = 20 \log_{10} \left(\frac{x_{\max}}{\sqrt{\text{MSE}}} \right) \tag{4.2}$$

where $x_{\max}$ is the maximum pixel value of the image, and MSE is the Mean Squared Error between the original and reconstructed images given by:

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \left(x_{ij} - \hat{x}_{ij}\right)^2 \tag{4.3}$$

A higher PSNR value indicates better image quality and lower reconstruction error. PSNR is effective in measuring the fidelity of the reconstruction in terms of the pixel intensity levels.

4.1.3 Structural Similarity Index (SSIM)

Structural Similarity Index (SSIM) [1] is a perceptual metric that assesses the visual impact of differences between the original and reconstructed images. Unlike NMSE and PSNR, SSIM considers changes in structural information, luminance, and contrast. SSIM is defined as:

$$\text{SSIM}(x, \hat{x}) = \frac{(2\mu_x\mu_{\hat{x}} + C_1)(2\sigma_{x\hat{x}} + C_2)}{(\mu_x^2 + \mu_{\hat{x}}^2 + C_1)(\sigma_x^2 + \sigma_{\hat{x}}^2 + C_2)} \tag{4.4}$$

where μ_x and $\mu_{\hat{x}}$ are the mean intensities, σ_x^2 and $\sigma_{\hat{x}}^2$ are the variances, and $\sigma_{x\hat{x}}$ is the covariance of the original and reconstructed images, respectively. C_1 and C_2 are constants to stabilize the division. SSIM values range from 0 to 1, with 1 indicating best structural similarity.

4.2 Convolution results

In this section, we analyze the results of three convolution schemes to validate the accuracy of the proposed convolution scheme using different dataset scenarios, specifically 2D and 3D image data. The schemes are implanted on Pyxu [2]. Furthermore, we investigate the relationship between the accuracy of the convolution schemes, computational cost, and the energy of the PSF. Based on this analysis, we aim to develop a method to select optimal parameters for the convolution scheme, balancing accuracy and computational efficiency.

4.2.1 Convolution with 2D images

We performed 3 convolution schemes using a 2D sinusoidal Siemens star image of size 1024×1024 and a 2D simulated PSF of the same size [3].

Figure 4.1 shows the separated PSF components $\mathbf{h}_c$ and $\mathbf{h}_s$, and the results of the convolution $\mathbf{y}_{\text{ref}}$ and $\mathbf{y}_{\text{multi}}$ for $r = 21$ and $M = 2$. From the results, there are no aliasing effects or artifacts present in $\mathbf{y}_{\text{multi}}$. Numerical results of $\mathbf{y}_{\text{ref}}$, $\mathbf{y}_{\text{c}}$, and $\mathbf{y}_{\text{multi}}$ are shown in Table 4.1

From the table, although $\mathbf{y}_{\text{c}}$ achieved the lowest peak memory requirement and time usage, it is not accurate, with an NMSE of 10.99%. With downsampling parameter

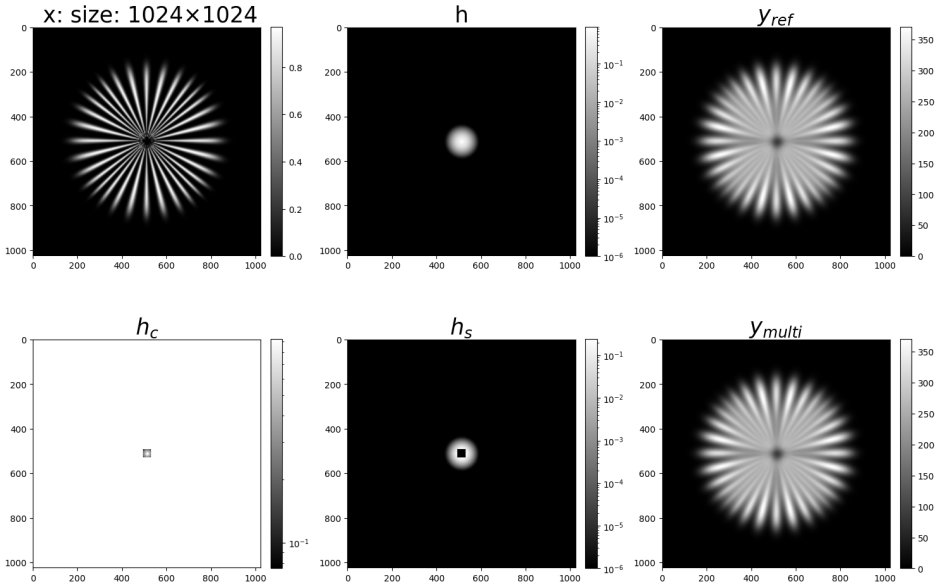


Figure 4.1: Results of 2D convolution

Method	r	M	Energy[%]	Time[s]	Peak Memory[Mb]	NMSE[%]	SSIM	PSNR[dB]
$\mathbf{y}_{\text{ref}}$	-	-	100.000	0.226	405.156	-	-	-
$\mathbf{y}_{\text{c}}$	21	-	92.349	0.030	278.08	10.99	0.936	20.194
$\mathbf{y}_{\text{multi}}$	21	4	92.349	0.092	229.922	0.0087	0.996	51.323
$\mathbf{y}_{\text{multi}}$	21	2	92.349	0.111	230.109	5.6×10^{-5}	0.999	73.120

Table 4.1: Performances with different convolution parameters

M=2, $\mathbf{y}_{\text{multi}}$ decreases the cost of computation by approximately half while maintaining high accuracy. Additionally, using a larger downsampling factor does not bring significant improvement in computational cost.

Figure 4.2 shows the energy contained in $\mathbf{h}_c$ as r increases. It further demonstrates that most of the energy on PFS is concentrated in a small portion of the PSF. In this case, when $r = 50$, $K_c = 101$ and $\mathbf{h}_c$ takes approximately 9.86% of the size of PSF but contains 99.99% of the energy.

To determine an appropriate range for the decimation factor M, the accuracy of $\mathbf{y}_{\text{multi}}$ as a function of M is illustrated in Figure 4.3. Here $r = 20$, and $\mathbf{h}_c$ contains 90% of PSF's energy. As M increases, the accuracy of $\mathbf{y}_{\text{multi}}$ declines. However, for $M = 2$ and $M = 4$, high accuracy is still maintained, with NMSE remaining below 0.0023% and SSIM approaching 1.

Figure 4.4 shows the accuracy metrics of 3 convolution schemes over r and energy contained in $\mathbf{h}_c$. Results of NMSE and SSIM are shown in log scale. Accuracy increases significantly when the energy of $\mathbf{h}_c$ is between 80% to 100%. Thus, choosing r such that the energy of $\mathbf{h}_c$ is between 80% to 100%, as shown in Figure 4.2, suggests that selecting r between 15 to 30 would achieve high accuracy. Since $\mathbf{y}_{\text{multi}}$ schemes exhibit better accuracy compared to $\mathbf{y}_{\text{c}}$, this highlights the performance gains achieved by

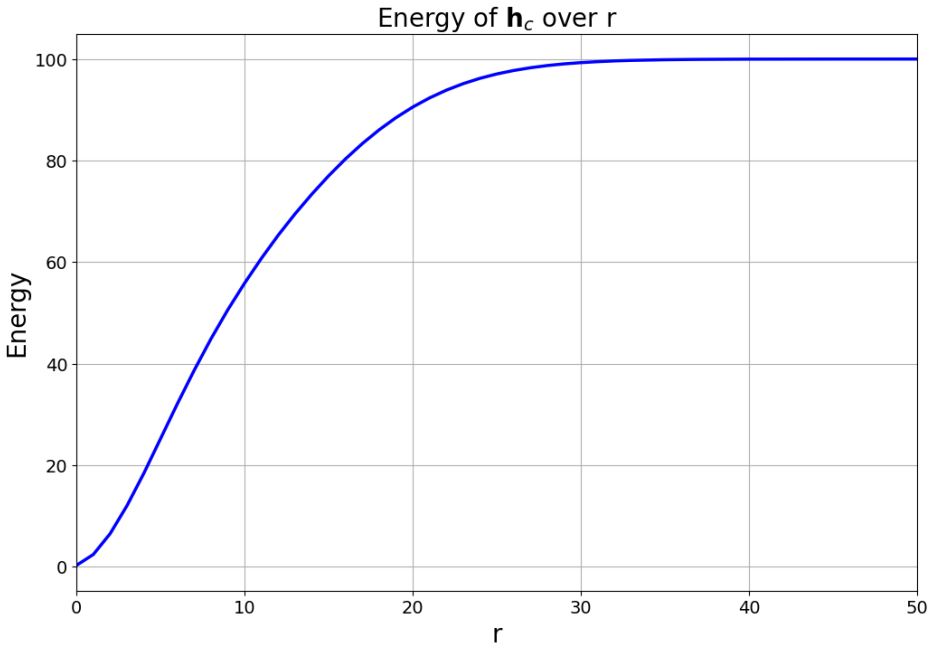


Figure 4.2: Energy of $\mathbf{h}_c$ over r

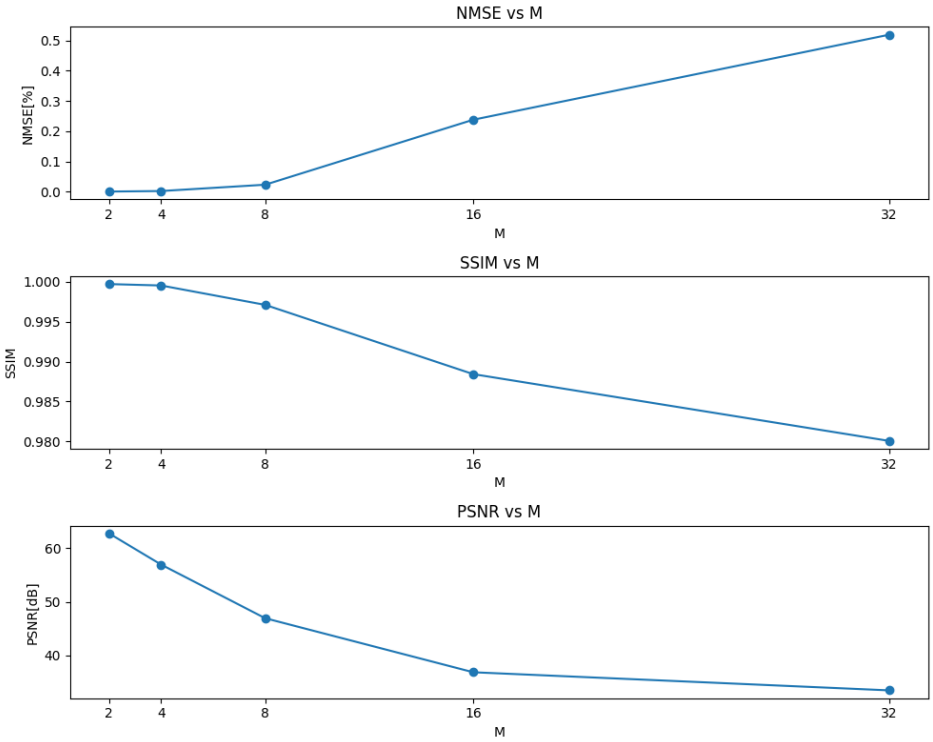


Figure 4.3: Performance metrics of $\mathbf{y}_{\text{multi}}$ for different decimation factors M

incorporating information from the sidelobes $\mathbf{h}_s$.

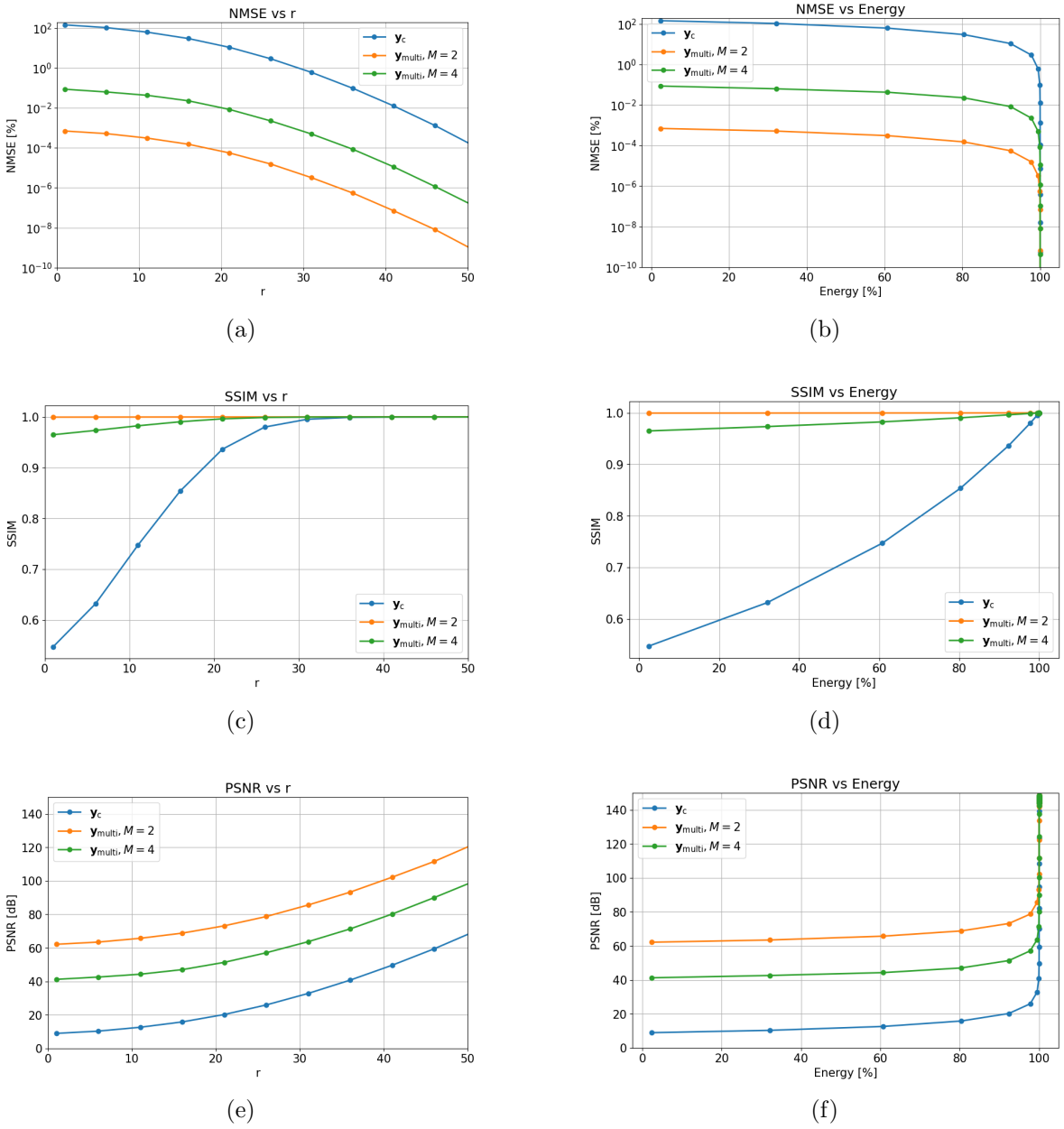
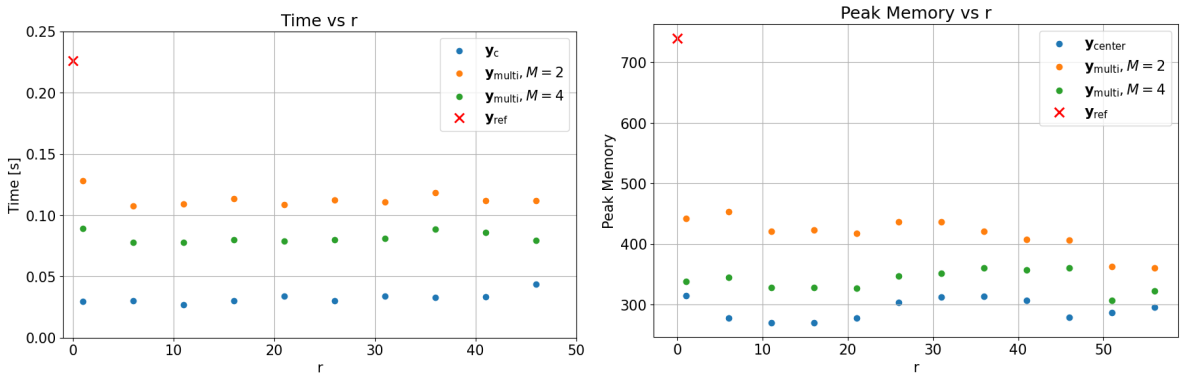


Figure 4.4: Accuracy of convolutions schemes over r

Figure 4.5 shows the computation time and the peak memory over r. These results align with the theoretical analysis in Section 2.3.1. Both time and memory requirements decreased significantly compared to $\mathbf{y}_{\text{ref}}$. However the reduction is less than M^d folds, which could be due to the computation overhead.

Additionally, with $\mathbf{y}_c$ and $\mathbf{y}_{\text{multi}}$ both computation time and peak memory do not vary much with changes in r. This matches the theoretical estimation of the complexity and peak memory. Both of them depend on the size of the image N and the downsampling factor M, but not on r.



(a) (b)

Figure 4.5: Computation cost over r

The theoretical and real peak memory results with different data sizes N are shown in Figure 4.6. From the Figure, the theoretical results and the experimental ones follow a similar trend, but there is an overhead for experimental peak memory results.

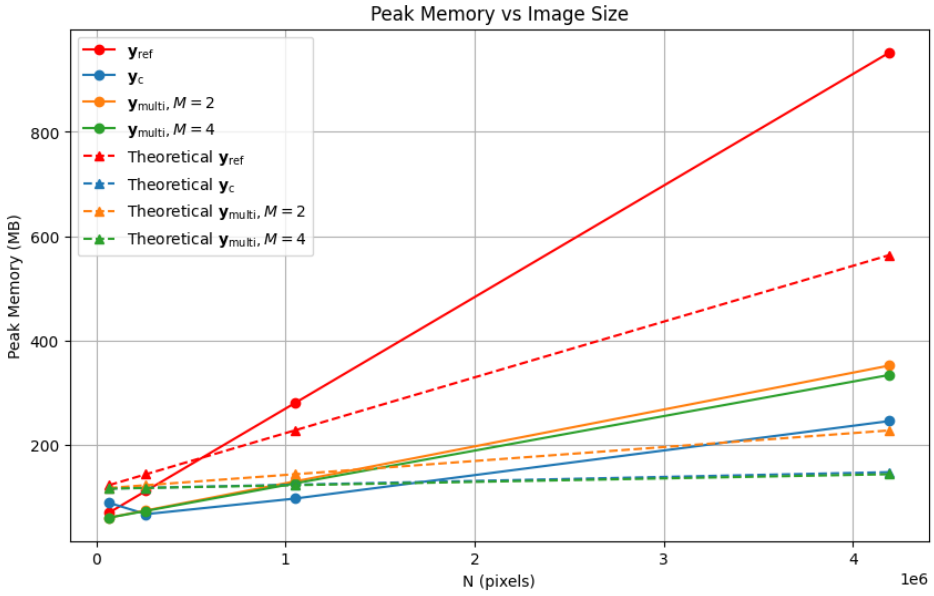
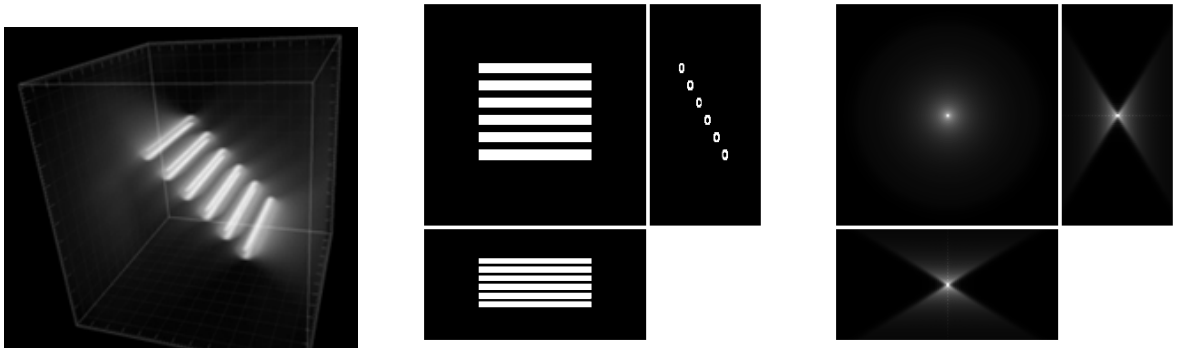


Figure 4.6: Peak memory results with different data size

4.2.2 Convolution with 3D image

The dataset is shown in Figure 4.7a. It consists of synthetic 3D data featuring six parallel hollow bars, and a theoretical microscopic PSF (Figure 4.7c). The image size is 256x256 pixels with 128 z-slices.

The convolution results are shown Figure 4.8. According to the result, it shows no aliasing or artifacts.



(a) 3D hollow bars data

(b) Maximum intensity projection of 3D hollow bar

(c) Maximum intensity projection of 3D PSF

Figure 4.7: 3D hollow bar data and PSF

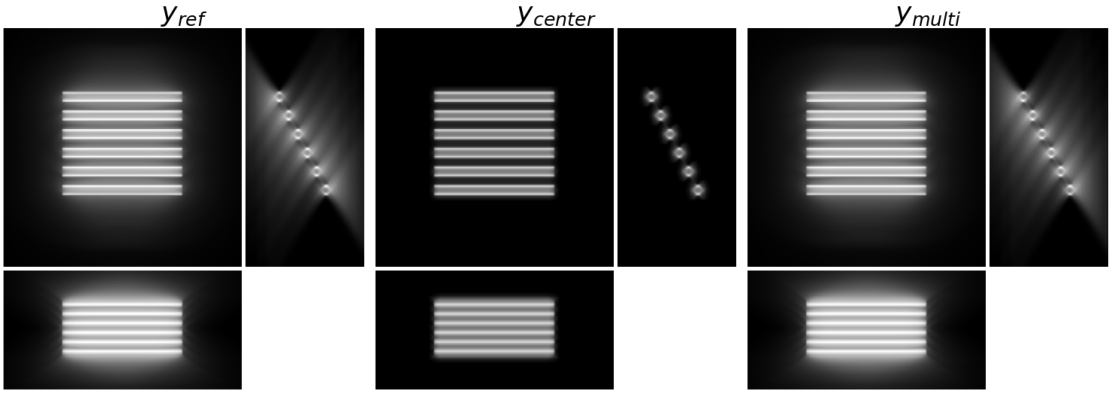
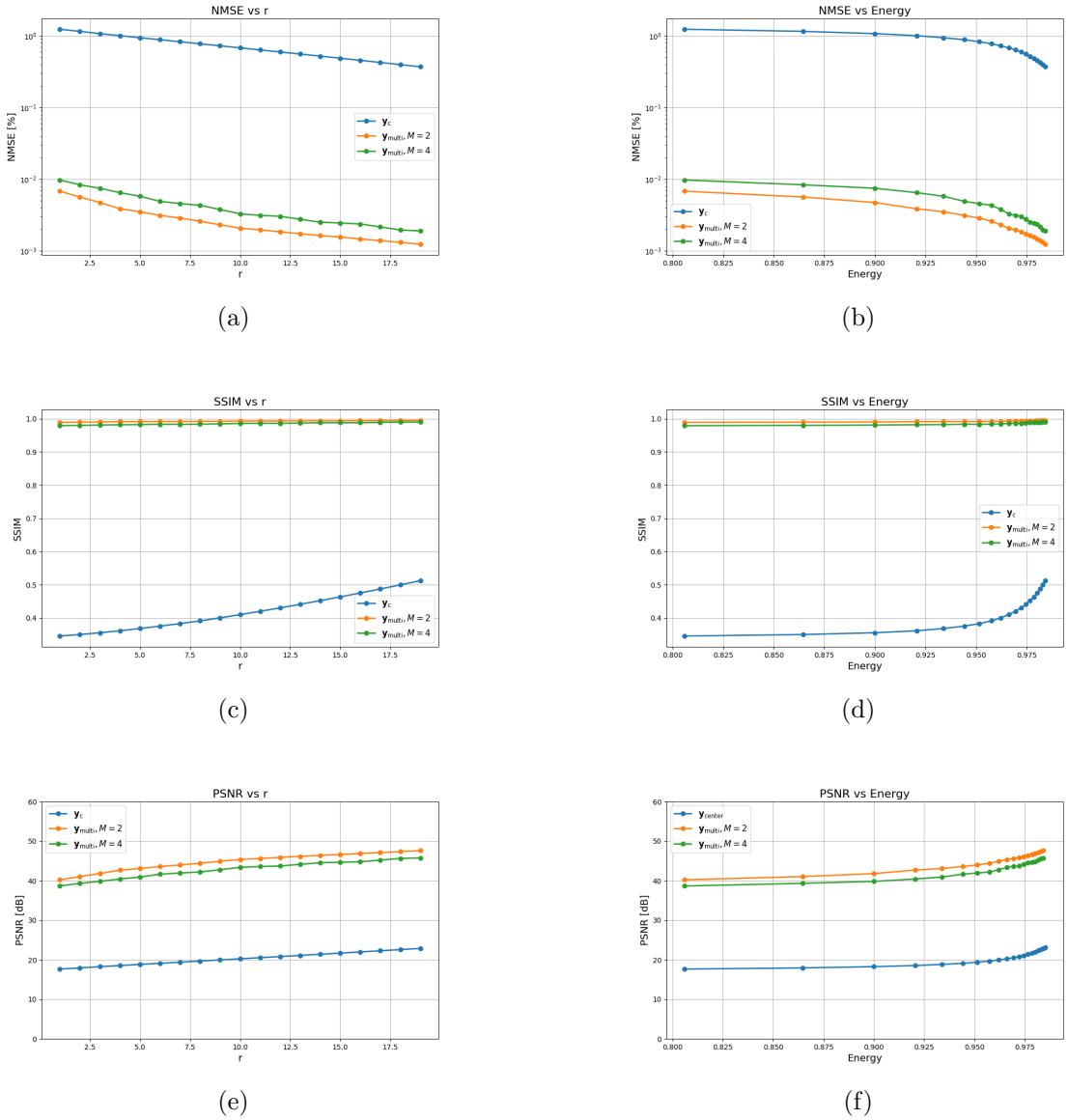


Figure 4.8: Convolution results of 3D hollow bar data

To simplify the analysis, set $\alpha = \frac{FWHM_x}{FWHM_z}$ and $r_x = \alpha r_z$. Here, $FWHM_z$ and $FWHM_x$ represent the full width at half maximum of the PSF along the z and x directions, respectively. Based on this setting, when r is increased in the x and y directions, r_z in the z direction increases simultaneously. The performance with varying r is shown in Figure 4.9. The overall results align with the 2D performance results. They also demonstrate that by keeping the side slopes part $\mathbf{h}_s$, the $\mathbf{y}_{\text{multi}}$ achieves higher performance increases.

The computational cost is illustrated in Figure 4.10. Both $\mathbf{y}_{\text{multi}}$ and $\mathbf{y}_{\text{c}}$ reduced the computational cost by approximately threefold in terms of both time and memory. However, this reduction is less than the theoretical prediction. One possible explanation for this discrepancy is memory overhead, suggesting that there is still potential for further optimization to minimize memory usage.

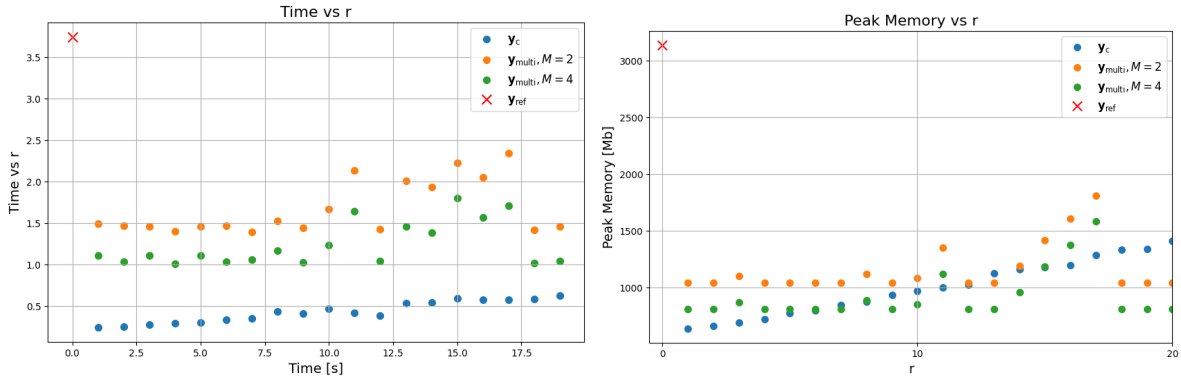


(a) (b) (c) (d) (e) (f)

Figure 4.9: Accuracies of convolution schemes with 3D images,varying r and $r_z = \alpha r$

4.3 Deconvolution

After validating the proposed convolution scheme's ability to achieve high accuracy and reduce computational costs, we aim to evaluate its performance within a deconvolution algorithm. For this purpose, we implemented the Richardson-Lucy deconvolution algorithm using the $\mathbf{y}_{\text{multi}}$ convolution scheme. The method was initially tested on simulated 3D data, followed by real-world data.

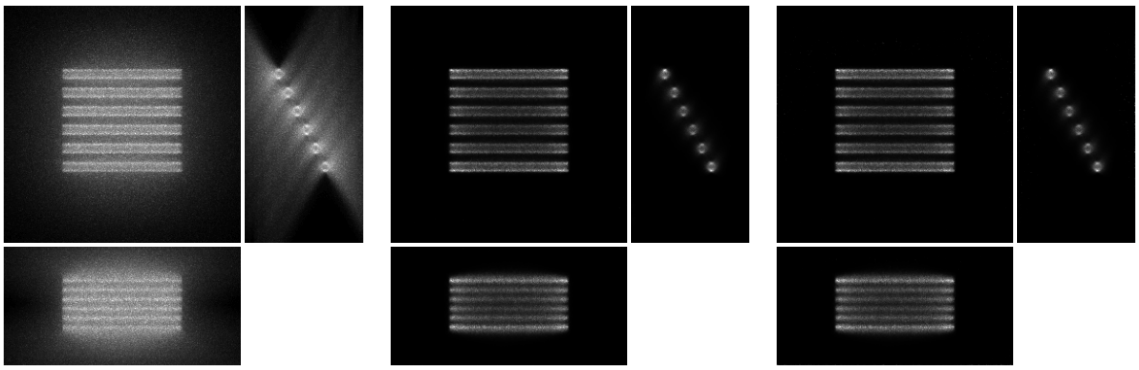


(a) (b)

Figure 4.10: Computation cost of 3D image data over r

4.3.1 3D simulation data

Figure 4.11 and Table 4.2 show deconvolution results after 30 iterations of Richardson-Lucy algorithm. Performance with similar NMSE and PSNR values was achieved with half of the time.



(a) y (b) $\hat{x}_{\text{ref}}$ (c) $\hat{x}_{\text{multi}}$

Figure 4.11: Comparison of y, $\hat{x}_{\text{ref}}$, and $\hat{x}_{\text{multi}}$

Methods	Iterations	Computation Time (s)	NMSE (%)	PSNR (dB)
$\hat{x}_{\text{ref}}$	30	129.17	68.5204	26.69
$\hat{x}_{\text{multi}}$	30	54.63	68.4197	26.60

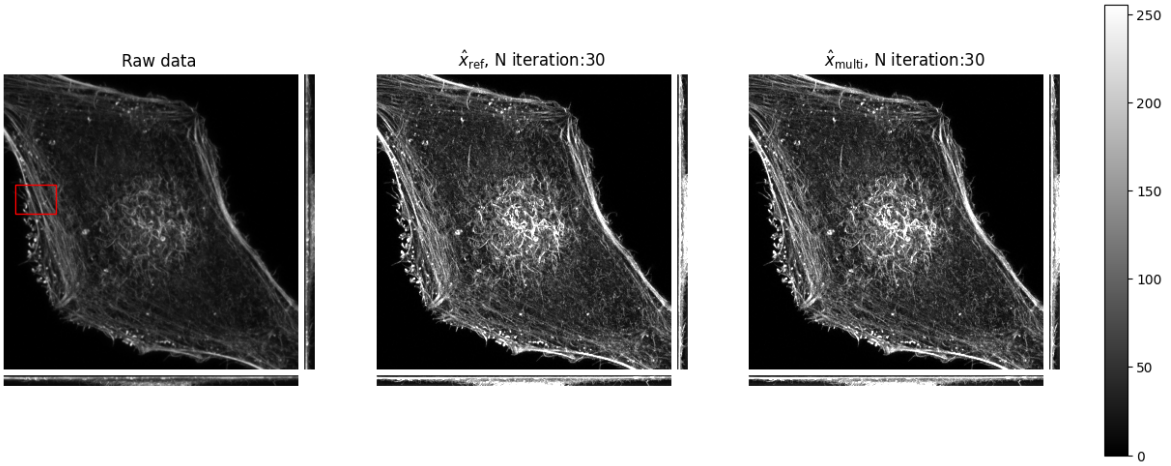
Table 4.2: R-L deconvolution results

4.3.2 Deconvolution with real data

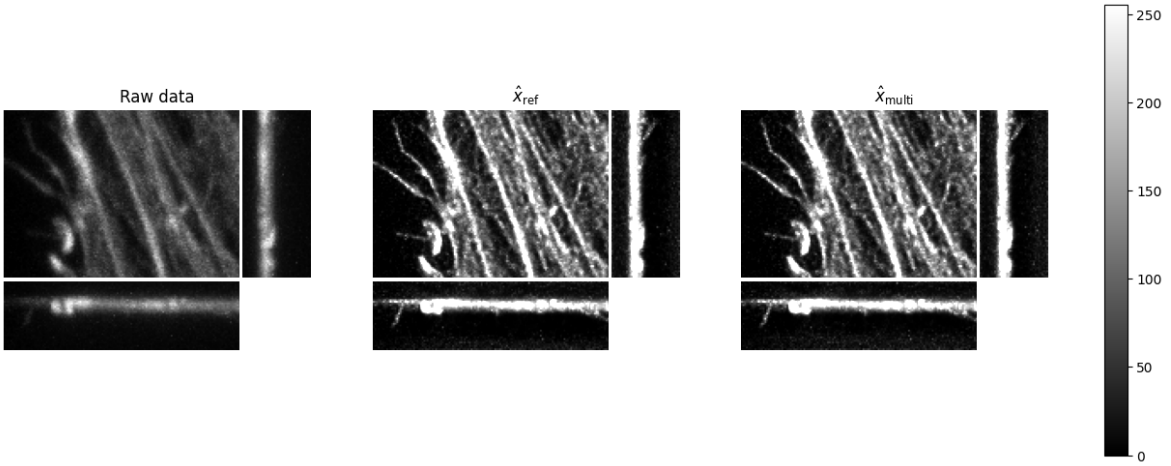
The deconvolution test dataset from [4] consists of HeLa cells stained for action, acquired on a confocal Zeiss LSM710 with a theoretical PSF generated with a PSF gen-

erator. The image size is 1024×1024 with 41 z-stacks.

Since no ground truth is available, we cannot apply the metrics mentioned above. Instead, we compare the results visually and consider the deconvolution result $\hat{\mathbf{x}}_{\text{ref}}$ as reference. The results are shown in Figure 4.12. After 30 iterations, the resolution of the images increases, with sharpened details. The NMSE between $\hat{\mathbf{x}}_{\text{ref}}$ and $\hat{\mathbf{x}}_{\text{multi}}$ is 0.13%.



(a)



(b)

Figure 4.12: Deconvolution results with 3D real data

To further examine the computational cost, the peak memory usage for a single convolution operation with this dataset across different convolution methods is presented alongside the accuracy metrics (NMSE) in Figure 4.13. For convolution using the full PSF, the peak memory reaches approximately 16.14 GB, which exceeds the memory limit of many laptops, making it impractical to run the deconvolution algorithms in such environments.

In contrast, multiresolution methods offer a significant reduction in memory usage. For

instance, with a decimation factor of $M = 2$, the peak memory drops to around 6.93 GB, while still achieving a very low NMSE of 0.013%.

As the decimation factor increases, the memory consumption continues to decrease. For $M = 4$, the peak memory usage is further reduced to 5.82 GB, with a minor increase in NMSE to 0.026%. When $M = 8$, the peak memory is minimized to 5.67 GB, but this reduction comes at the cost of a noticeable drop in accuracy, with NMSE rising to 0.505%. Additionally, the convolution using $\mathbf{h}_c$ requires even less memory, at 4.90 GB, but suffers from reduced accuracy, with NMSE reaching 0.987%.

These results highlight the benefits of our multiresolution convolution scheme. It significantly reduces peak memory consumption while maintaining high accuracy, demonstrating its practicality for memory-constrained environments.

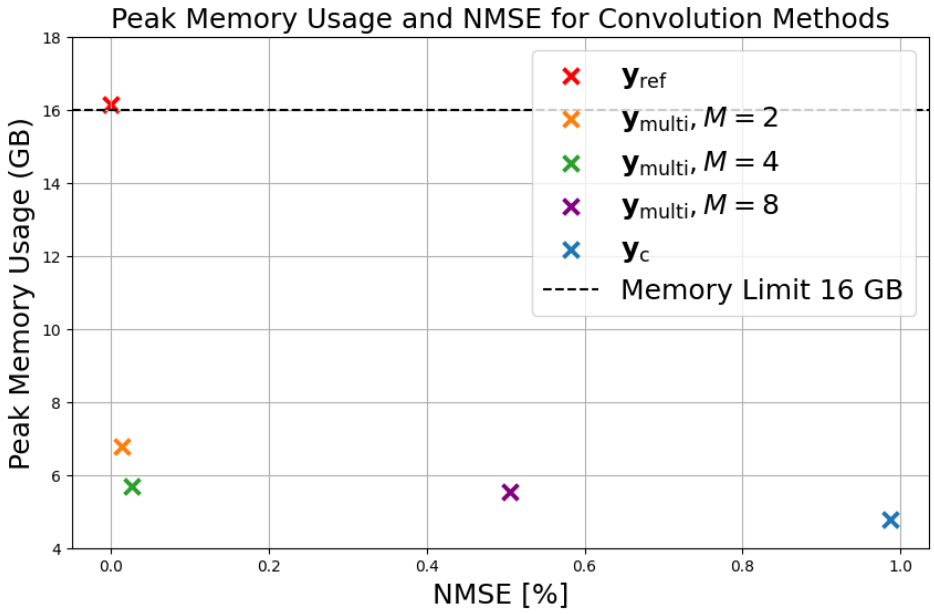


Figure 4.13: Peak memory usage and NMSE for different convolution methods

References

[1] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[2] M. Simeoni, S. Kashani, J. Rué-Queralt, and P. Developers, *Pyxu-org/pyxu: Pyxu*. DOI: 10.5281/zenodo.4486431. [Online]. Available: https://doi.org/10.5281/zenodo.4486431.

[3] D. Sage, L. Donati, F. Soulez, *et al.*, "Deconvolutionlab2: An open-source software for deconvolution microscopy," *Methods*, vol. 115, pp. 28–41, 2017.

[4] R. Guiet, *Deconvolution test dataset*, version v0, Zenodo, Jul. 2021. DOI: 10.5281/zenodo.5101351. [Online]. Available: https://doi.org/10.5281/zenodo.5101351.

5 Conclusions and Future work

5.1 Convolution

This project introduced a multi-resolution convolution scheme that addresses the computational challenges in microscopy image processing, particularly for large-scale 3D data. The motivation for this approach stems from the significant computational load imposed by convolutions, which are essential for iterative deconvolution algorithms used in microscopy. By leveraging the structure of the Point Spread Function (PSF), our method segments the PSF into a central, energy-dense part and its sidelobes, and implements multi-resolution signal processing methods. This allows for downsampling of the latter to reduce computational complexity.

The main conclusions of the study are as follows:

The proposed multi-resolution convolution method achieves a significant reduction in memory usage and computation time without sacrificing accuracy. The technique is highly effective even for large datasets, both in 2D and 3D imaging. The method exploits the fact that most of the PSF energy is concentrated in a small central region, enabling efficient downsampling of the sidelobes while maintaining overall image quality.

In comparative performance tests, the multi-resolution convolution consistently outperformed traditional full PSF-based convolution methods, offering similar levels of accuracy (based on NMSE, PSNR, and SSIM metrics) while halving computational costs. When optimally chosen, $M = 2$ or $M = 4$, the scheme delivered significant improvements in efficiency with minimal loss of precision.

The versatility of the approach was demonstrated across different microscopy modalities, making it broadly applicable to various imaging techniques where the PSF structure allows for similar segmentation. By separating the central, high-energy part of the PSF from the less critical sidelobes, the method can be extended to higher-dimensional datasets and other imaging systems beyond microscopy.

The effectiveness of this approach was further demonstrated in its integration into the Richardson-Lucy deconvolution algorithm. Both synthetic and real microscopy data showed enhanced image quality, with faster processing times, making this method particularly suitable for practical applications in biological imaging where large datasets and iterative deconvolution are common.

5.2 Future work

There are several promising directions for future research and improvement of the multi-resolution convolution method:

First, implementing polyphase decomposition during the upsampling and interpolation steps can bring further improvement. This could reduce computational cost even further by optimizing the multiresolution scheme, leading to more efficient convolution operations.

Another important potential improvement is optimizing memory usage. Although the current method achieves reductions in memory consumption, there is still a gap between the experimental results and the theoretical minimum. Further refinement of the algorithm could bridge this gap. Additionally, hardware acceleration could be implemented by leveraging GPUs and parallelizing the convolutions, which would further enhance processing speeds and efficiency.

Enhancing the PSF segmentation strategy is another potential improvement. Currently, a fixed radius is used to separate the central part of the PSF from its sidelobes. The PSF could be divided into multiple regions and downsampled with varying decimation factors, depending on the energy distribution in each section, further optimizing the process.

Additionally, establishing a theoretical lower bound for the errors in the multiresolution convolution scheme would be valuable. Determining such a bound would not only serve as a benchmark for the algorithm's accuracy but also offer insights on choosing the parameters based on the hardware's memory condition and the desired accuracy.

Finally, extending the multiresolution convolution scheme beyond deconvolution to other algorithms and application domains represents an exciting direction for future research. For other microscopy deconvolution algorithms that require convolution as part of their implementation, the multiresolution scheme can be applied to reduce computational costs. Additionally, this approach could be adapted to benefit other areas of image processing where convolution is used, particularly when the kernel has a structure similar to the PSF.

In summary, this project presents a solution to the out-of-memory challenges in microscopy image processing by providing a convolution scheme that balances computational efficiency with high accuracy. With further optimization and development, the multi-resolution convolution scheme holds promise for a wide range of applications in both microscopy imaging and broader image processing fields.