PSF SAMPLING IN FLUORESCENCE IMAGE DECONVOLUTION

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Eric Inman

March 2023

COMMITTEE MEMBERSHIP

TITLE:	PSF Sampling in Fluorescence Image Deconvolution
AUTHOR:	Eric Inman
DATE SUBMITTED:	March 2023
COMMITTEE CHAIR:	Jonathan Ventura, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Maria Pantoja, Ph.D. Professor of Computer Science
COMMITTEE MEMBER:	Guy Hagen, Ph.D. Research Associate, UCCS BioFrontiers Center

ABSTRACT

PSF Sampling in Fluorescence Image Deconvolution

Eric Inman

All microscope imaging is largely affected by inherent resolution limitations because of out-of-focus light and diffraction effects. The traditional approach to restoring the image resolution is to use a deconvolution algorithm to “invert” the effect of convolving the volume with the point spread function. However, these algorithms fall short in several areas such as noise amplification and stopping criterion. In this paper, we try to reconstruct an explicit volumetric representation of the fluorescence density in the sample and fit a neural network to the target z-stack to properly minimize a reconstruction cost function for an optimal result. Additionally, we do a weighted sampling of the point spread function to avoid unnecessary computations and prioritize non-zero signals. In a baseline comparison against the Richardson-Lucy method, our algorithm outperforms RL for images affected with high levels of noise.

ACKNOWLEDGMENTS

A very special thanks to:

- My Mom and Dad who constantly supported me through everything. I love you guys!
- All the professors that helped me along the way, especially Dr. Ventura. Thank you for being patient with me. I couldn't have done this without you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Fluorescence Microscopy

1.1.1 Overview

Fluorescence microscopy is an imaging technique used in light microscopes that uses a strong light to excite fluorophores which in turn, emit a low energy light that gets used to produce a magnified image. Fluorescence microscopes are much the same as conventional microscopes except they require a much higher intensity light source and a dichroic mirror to capture the image [7]. It is widely used by scientists to observe the localization of molecules within cells, and of cells within tissues. Since the light beam penetrates the full depth of the sample, researchers are easily able to image intense signals or perform co-localization studies with multicolored fluorophores [8].

In this type of microscopy, cell samples are very sensitive to the amount of light that they are exposed to. If the cells are exposed to light for too long or the strength of the illumination is too great, the sample can experience great damage through phototoxicity or even undergo changes in its physiological structure [18]. If exposed to too little light, the resulting images of the cells can be unreadable to biologists. Optimizing these images is vastly important because they lend great potential to understanding cellular processes and structures. It provides a window into the physiology of living cells at sub-cellular levels of resolution. This is similar to looking at a systems level view of a living cell, which can provide valuable insight to phenomenon like ion transport or metabolism.

1.1.2 Widefield vs. Confocal Microscopes

There are two major groups of microscopes that can be used in fluorescence imaging: widefield microscopes and confocal microscopes.

In conventional widefield systems, 3D information from the object is recorded in a set of 2D images of different in-focus planes of the object. The issue with this system is that light emitted from out-of-focus regions gets collected just the same as light from in-focus regions [27]. Because of this, the resulting 3D image contains both in-focus plane and out-of-focus information which limits contrast and prevents clear identification of structures of interest.

In confocal microscope systems, the out-of-focus light is deflected through the use of a pinhole in front of the detector which improves the lateral resolution. The high resolution comes at the cost of a lower SNR (signal-to-noise ratio) in the 3D image because only light that passes through the pinhole is collected [27].

Our project will focus on fluorescence imaging done with widefield microscopes because we want to maximize the signal in the image while correcting for any sources of image degradation that are present. Regardless, our method should be applicable to any type of blurred image as long as there is an associated point spread function.

1.2 Deconvolution Methods

1.2.1 Overview

Deconvolution is a computational technique that is applied to digital imagery to compensate for the optical limitations of the imaging instrument by reducing out-of-focus blurring. The most commonly used deconvolution algorithms can be divided into two main classes: deblurring and image restoration. Deblurring algorithms are usually two-dimensional and work by applying a deblurring operation on each 2D plane in a three-dimensional image stack [3]. On the other hand, image restoration algorithms are three-dimensional in that they operate simultaneously on every pixel in the stack.

1.2.2 Deblurring Algorithms

Some common deblurring algorithms include nearest-neighbor, multi-neighbor, and unsharp masking. As mentioned before, these algorithms are two-dimensional and operate on each plane in a three-dimensional image stack. While these algorithms are computationally inexpensive because of their relatively simple calculations, they have a few disadvantages. One is that noise is added together from several planes. Another problem is that these algorithms remove blurred signal which reduces the overall signal levels of the image. Lastly, certain features can be sharpened in planes where they don't belong, resulting in altered feature positions [3]. Our project will focus on an image restoration algorithm.

1.2.3 Image Restoration

Image restoration deals with blur in image stacks as a three-dimensional problem. Instead of getting rid of blur by subtracting it, these algorithms attempt to reassign the light to its proper in-focus location [3]. To do this, the collected images are modelled as a convolution between the true 3D object and the point spread function used in the imaging system.

A point spread function is a three-dimensional diffraction pattern of light emitted from an infinitely small point source in the specimen and transmitted to the image plane through a high numerical aperture [5]. When light is emitted from such a point object, a fraction of it is collected by the objective and focused at a corresponding point in the image plane. However, the objective lens does not focus the emitted light to an infinitely small point in the image plane. Rather, light waves converge and interfere at the focal point to produce a diffraction pattern of concentric rings of light surrounding a central, bright disk, when viewed in the x-y plane [27]. PSFs can be theoretically or empirically obtained. Theoretical PSFs are defined by utilizing a mathematical model of diffraction, while empirical PSFs can be obtained by acquiring a three-dimensional image of a fluorescent bead.

Deconvolving the collected image with the point spread function should restore the true image. The advantage of formulating the problem in this way is that we can compute convolution operations on large matrices simply by using the Fourier space. By converting the PSF and the raw image into their Fourier counterparts, we can get the real image by dividing the raw image by the PSF, and then transform the resulting image back into the three-dimensional coordinate space [3].

Two examples of image restoration techniques are Richardson-Lucy deconvolution and Wiener deconvolution. Wiener deconvolution is also known as an inverse filter algorithm because it uses division in the Fourier space to undo the convolution. Richardson-Lucy is known as an iterative restoration algorithm because it continually tries to minimize the difference between an estimated image and the blurry image [3]. RL is one of the most popular deconvolution algorithms because it's relatively simple, improves the likelihood that the next iteration is correct, and assumes a Poisson noise distribution. Both of these algorithms are limited by noise amplification, but for different reasons. Because Wiener filtering uses division in the Fourier space, small noise variations in the Fourier transform can be amplified with a small denominator in the division operation. For Richardson-Lucy, repeated convolution operations can introduce high frequency noise [4]. An artifact known as ringing can also occur, but both this and noise can be mitigated to an extent if some assumptions are made about the object structure.

1.3 Motivation

Algorithms that work in the Fourier space make use of the fast Fourier transform to efficiently compute convolutions between the point spread function and the images. Ultimately in our project, we wanted to explore the possibility of using different representations such as NeRF [23] or TensoRF [16]. The problem with switching the representation like this is that we lose the ability to compute fast Fourier transforms. Because of this, we have to compute the convolution directly in the spatial domain, which can become very expensive. The need to minimize unnecessary computations is what drives the idea of sampling the point spread function. Not only do we want to minimize these computations, we also want to make sure that using a subset of the point spread function is enough to achieve good resolutions.

Chapter 2

RELATED WORKS

There are several related works that make significant contributions to this field, and the ideas behind a few are used in this project. These works focus on two main areas: neural networks and optimizations. The first group deals with any related research that incorporates neural networks either through resolution improvements from changes in network architectures or archetypes, or from using structures that can be implemented through neural networks. Optimization works either make new assumptions about the imaging system that lead to resolution improvements, or they utilize new data structures to optimize calculations and reduce deconvolution runtimes. First, the neural networks are discussed followed by the optimization works.

2.1 Neural Networks

2.1.1 Network Archetypes

The work done by Weigert et al. presents a solution for the problem of missing training data in fluorescence microscopy. They proposed using deep learning in what is called a content-aware restoration network to produce resulting images that would be unobtainable in normal circumstances [29]. By training this convolutional network, they were able to achieve a reliable degree of restoration on most fluorescence microscopy images. They also showed concrete examples where microscopy images were restored with 60-fold fewer photons used during acquisition, and how near isotropic resolution can be achieved with up to 10-fold under-sampling along the axial direction.

One of the more interesting works done fairly recently is by Lee et al. where they present a 3-way spatially constrained cycle-consistent generative adversarial network for deconvolving volumes [19]. The GAN gets trained along the xy, yz, and xz sections of the volume to be able to restore the microscope images. Their method is also blind meaning that the algorithm doesn't know anything about the point spread function. Their results showed good improvements in both blurry and noisy images.

2.1.2 Neural Representations

One of the most widely used structures in 3D scene reconstruction is the Neural Radiance Field (NeRF) [23], which was expanded upon by Ma et al. Their method, called Deblur-NeRF, recovers a sharp NeRF from blurry input which allows for a sharper scene reconstruction in blurry images [21]. Their method showed very good results for both synthetic and real-world examples.

Another variation to the original NeRF project is mip-NeRF, which extends the original neural radiance field to reduce objectionable aliasing artifacts. The results of their experiments showed a 17% average reduction to error rates as well as a 7% speed boost to performance [14]. There are many other variations of the NeRF structure which can be utilized for this kind of problem.

2.1.3 Miscellaneous

A very good neural network based reconstruction algorithm comes from the work of Zhong et al. With this, they were able to reconstruct the 3D structure of a protein from unlabeled 2D cryo-electron microscopy images [31]. Their reconstructed images were comparable to state-of-the-art methods at the time. Their method was interesting in that it encoded structures in the Fourier space using coordinate-based variational autoencoders.

2.2 Optimizations

2.2.1 Microscope System Assumptions

Zhao et al. makes a significant improvement to the resolution of fluorescence microscopy images through the use of a sparse deconvolution loss function. This function uses sparsity and continuity as the two main general features of the fluorescence microscope [30]. Sparsity balances the extraction of high-frequency information, while the Hessian matrix continuity reduces artifacts and increases robustness at the cost of reduced resolution. Their results showed a two-fold increase in spatial resolution within 3D images. Their implementation also uses an explicit representation of the volume, which we incorporate in our work.

Most works in this area make the assumption that the point spread function is space invariant to make the deconvolution process a bit easier. This work, done by Chen et al., was really interesting in that they assume the microscopy system is space-variant. From this, they developed a method to estimate space-variant point spread functions and use them in deblurring algorithms, which showed better signal-to-noise ratios and higher image qualities in both simulated and real data [17].

2.2.2 Performance Improvements

Another related work to this project comes from Muller et al. Their work focuses on making the training and evaluation of neural network primitives less costly and time-consuming through the use of a multiresolution hash table. The structure allows for easy parallelization on modern GPUs which significantly improves training time without the cost of performance [24]. We don't implement this in our project, but it could be very useful for future work.

At the time this paper is written, there are no other works that investigate explicitly sampling the point spread function.

Chapter 3

DATA

3.1 Biomedical Imaging Group

All the data for this project comes from the website for the Biomedical Imaging Group. They pursue research on the development of new algorithms and mathematical tools for advanced processing of medical and biological images. The main topics they focus on are image reconstruction, multi-modal imaging, image analysis, and visualization [6]. They are also conducting research in the areas of mathematical aspects to imaging and application-oriented projects collaborating with medical and biology researchers. The website itself is host to a plethora of useful information ranging from recent research on medical imaging and signal processing to downloadable code that can be used in replicating work or conducting new research. For this project, we use two datasets that are publicly available on this website: the Simulation of 3D Microtubules and the C. Elegans Embryo datasets.

3.2 Test Dataset

The first dataset that we used in this project is the 3D microtubules dataset. It is a realistic-looking, synthetic 3D image representing a network of microtubules in a cell [26]. The images are 32-bit with dimensions $256 \times 128 \times 128$, and the set comes with the ground truth, the test volume which is the ground truth convolved with a point spread function with some added noise, and the point spread function that was used to generate the test volume. We use this dataset in our project to obtain real performance measurements to see how our method compares to other baseline methods like Richardson-Lucy.

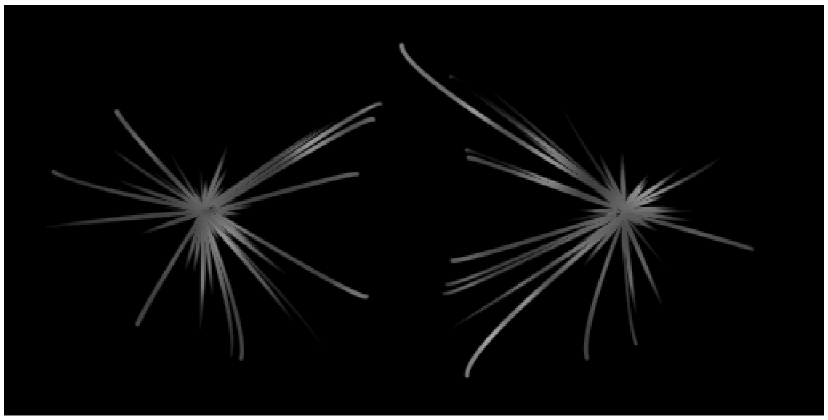


Figure 3.1: Example of the synthetic microtubules data.

3.3 Real-World Data

We want to make sure that this method not only works on simulated data, but also real-world data. For that, we use the C. Elegans Embryo dataset [1]. This is a real dataset composed of three stacks of images of a C. Elegans embryo. There are three kinds of structures that we can evaluate with this dataset. There are extended objects (the chromosomes in the nuclei), the filaments (the microtubules), and point-wise spots (a protein stained with CY3). For the purposes of this paper, we'll make a qualitative evaluation on all three of the embryo structures. The dataset also comes with a point spread function specific to each structure.

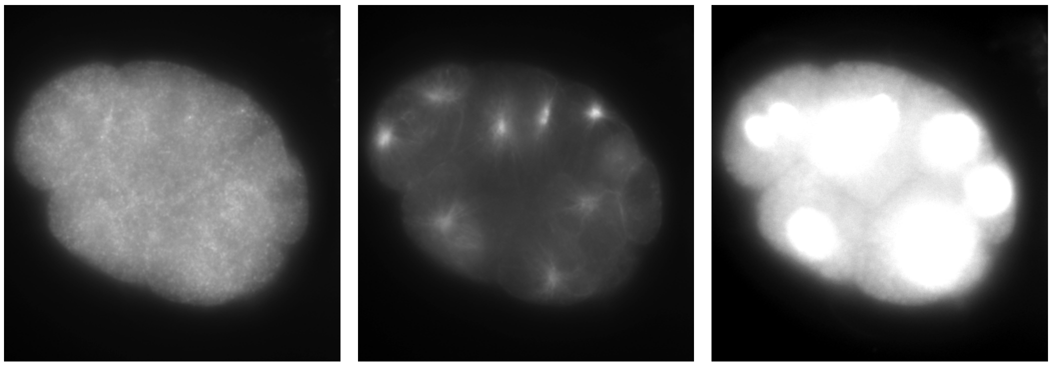

(a) (b) (c)

Figure 3.2: CY3 (a), FITC (b), DAPI (c) data examples.

3.4 Point Spread Functions

Our project uses a wide variety of point spread functions for different purposes. Most of them are created by us. The only point spread functions that aren't are the ones associated with the real-world data since we don't have a ground truth to convolve. We use three sets of PSFs: a set of non-isotropic PSFs, a set of Born and Wolf 3D optical model PSFs, and set of PSFs for the C. Elegans embryo. A lot of our PSFs feature different sizes. When we talk about PSF sizes in this context, we're talking about the 3D container that holds the PSF. We don't change any of the parameters that directly affect the point spread function such as the size of the numerical aperture or the wavelength of the light. So, the point spread function remains the same size, but the 3D container size does change. The non-isotropic PSF has two different sizes. The bigger one has size $31 \times 255 \times 255$ and the smaller one has size $7 \times 15 \times 15$. For the Born and Wolf PSFs, they were created using the PSF Generator plugin with default values for wavelength (610 nm), NA (1.4), pixelsize XY (100 nm), Z-step (250 nm), and refractive index immersion (1.5 ni). We did use various sizes of this PSF to test how the change in PSF size affects the algorithm. The real-world PSFs all have the same features except for the wavelength. They all feature a widefield type, a refractive index of 1.518 ni, and a numerical aperture of 1.4. The wavelengths are 477 nm, 542 nm, and 654 nm for FITC, DAPI, and CY3, respectively. Images for the PSFs can be found in Appendix A.

Chapter 4

IMPLEMENTATION

4.1 Libraries

4.1.1 TensorFlow

TensorFlow is one of the most popular python machine learning libraries to date. Created by Google and released in 2015, TensorFlow provides developers with an easy way to implement custom deep learning models, model tracking, and performance monitoring [13]. Under the hood, TensorFlow operates on what's called dataflow graphs. These are structures that describe how data moves through a graph or, for TensorFlow's purposes, a series of processing nodes. Not only can you create your own models, TensorFlow also gives you access to a lot of state-of-the-art models as well as pre-trained ones that you can make direct use of.

TensorFlow is the backbone of our entire project. We use it in all aspects of our project from defininING and training the model to sampling from the point spread function. We'll be using TensorFlow version 2.4.1, and we'll go into more detail about the model that we use it for in the Model section.

4.1.2 NumPy

NumPy is a Python library that excels in scientific computing. It provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays [10]. Some of these operations include shape manipulation, discrete Fourier transforms, basic linear algebra, basic statistical operations, and more. At the core of NumPy is the ndarray object. This object encapsulates n-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. Ndarrays are a fixed size at creation and are able to facilitate advanced mathematical operations more efficiently than Python's built-in lists. Because of theses advantages, a growing number of scientific and mathematical Python-based packages are using NumPy arrays. In order to efficiently use much of today's scientific/mathematical Python-based software, one must also be very comfortable with NumPy arrays.

As mentioned before, NumPy is the library for doing advanced mathematics in Python. It's another very important backbone of our project. We use it for defining coordinate meshes for the volume and the point spread function and to reshape ndarrays for our neural networks. Even TensorFlow makes use of some of the operations provided by NumPy. The version that we use in our project is 1.23.4.

4.1.3 PIL

Pillow is a fork of the Python Imaging Library that adds image processing capabilities to the python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats which provides a solid foundation for a general image processing tool.

PIL makes it very easy for us to read and write TIFF files which is the format that our data volumes are in. We use PIL version 8.4.0.

4.2 Model

This section is going to go into detail about the specifics of the model that we train to do the deconvolution. Specifically, we'll talk about the implementation of the explicit volumetric representation of the fluorescence samples as well as the details behind sampling the point spread function.

4.2.1 Explicit Representation

One of the more important decisions in this project is the choice of representing the volume that we're trying to deconvolve. Some implementations use neural radiance fields which make use of multilayered perceptrons to represent the volume. For our implementation, we chose to do an explicit volumetric representation following the work done by Zhao et al. Due to the sparse nature of these fluorescence images, we can get away with representing the volume explicitly using all zeroes and use our minimization function to update the locations in space that require more light. To do this, we make use of NumPy and the *linspace* function to create 3D meshes. One mesh belongs to the volume and consists of a 3D box of coordinates where each coordinate corresponds to the location of a pixel in the original volume. The other mesh belongs to the point spread function. This mesh is a bit different from the volume mesh. Instead of coordinates, the PSF mesh stores offsets from the origin of the PSF. These offsets are used to wrangle in points of light based on a specific pixel coordinate that supposedly belong to that location. Once the base meshes are created, we add padding around the entirety of the volume mesh so that when the edges of the volume are being convolved, we don't get an out of bounds error with the offsets, which eliminates the need for handling the edge cases. The values of the padding also begin as zeroes so as to not introduce new artifacts to the deconvolved volume. This mesh gets passed off to a custom TensorFlow model that preserves the shape of the padded mesh but turns all the coordinates into learnable parameters.

4.2.2 Optimizer and Learning Rate

In terms of learning, we use an Adam's optimizer with a continuous learning rate decay function. We use an initial learning rate of 0.05 with 2500 warmup steps where the initial rate is scaled by a smooth reverse cosine function of a decay multiplier that we have set to 0.1. The learning rate is eased back to the normal rate after the number of warmup steps. We also have a max number of learning steps set to 1,000,000. If the model does reach this, we cap the learning rate at 0.0005.

4.2.3 PSF-Sampling

The main contribution that we're testing is the sampling of the point spread function. There are two main purposes to experiment with sampling the PSF. The first idea is to minimize the number of calculations that have to be performed when we convolve the recovered image with the point spread function. With any point spread function, there are large sections of black areas that are zero-valued. These are areas where light does not disperse. We hypothesize that these regions end up being redundant calculations during a convolution process since they always result in zeroes.

The second idea was to test if the sample size of the point spread function could lead to improved deconvolution results compared to certain baseline methods. In order to implement this, we enlisted TensorFlow to do the sampling for us. Using the $tf.random.categorical$ function, we can take a specified number of samples from the point spread function using the log-probability of the PSF intensity values. The overall function that we're minimizing is the one that you typically see in most deconvolution works with a slight variation:

$$\arg\min_{x} \sum_{i=1}^{n} (f_i - (h_s * x)_i)^2 \tag{4.1}$$

where f is the recorded image, h_s is the sampled point spread function, and x is our estimated image. The asterisk denotes the convolution operation. We do this for all n pixels in the volume. We also choose a mean squared error loss function to account for Gaussian noise that can be introduced through the numerical aperture of the objective. This equation is written in one-dimension for simplicity. Our implementation of this is three-dimensional. The idea is that we sample from the point spread function once using the log-probabilities of the PSF values as weights for the positions. Once sampled, we gather pixels for every coordinate that may be representative of the actual light that takes place at that point. From there, we get a density measurement from all the collected pixels in our estimated volume and compare it to the convolved image. In essence, we're using the point spread function to gather locations where the light is most likely displaced and return it back to its original location through the density calculation. This sampling is performed once so every single pixel that appears in the batch technically gets convolved with the same subset of the point spread function. Our estimated volume gets updated after calculating how different it is from the collected image. Mathematically, we can show that using the average of a sample of random locations approximates the full convolution with the following formulas:

$$(h * x)_i \approx \frac{1}{N} \sum_{n} x_{i+o_n} \tag{4.2}$$

$$P(o_n = k) = h_k \tag{4.3}$$

Again, the equations are written in one-dimension for simplicity. Equation 4.2 approximates the result of a convolution at location i as an average of samples around that location. Equation 4.3 states that the samples are chosen according to the values of the PSF, which we assume already sum up to 1. As the number of samples approaches infinity, the sampled version approaches the actual convolution. Another thing to note is that we sample with replacement. This makes it so that the number of samples isn't dependent on the size of the point spread function and so that we're more partial to pixels that have a high probability of belonging to the original location.

Chapter 5

EXPERIMENTS

In total, we ran this method through five different experiments. For four of these tests, we calculate two metrics that are normally used to evaluate the quality of the images.

The first is called peak-signal-to-noise ratio (PSNR). PSNR is a ratio between the maximum possible value of a signal and the power of distorting noise that affects the quality of the image. The PSNR is calculated using this equation:

$$PSNR = 20 * \log_{10}(\frac{MAX_f}{\sqrt{MSE}}) \tag{5.1}$$

The higher the PSNR, the better the estimated image has been reconstructed to match the original image, which suggests a better reconstruction algorithm. The main limitation with this metric is that the comparison is strictly numerical in that it doesn't take into account any biological factors of the human vision system. The skimage python library already has this metric implemented so we simply make use of that.

The second metric is called structural similarity index (SSIM). This measurement is perceptual and quantifies image quality degradation caused by processing. Ideally after processing, we want the reconstructed image to the have the same structure as the ground truth. The higher the SSIM score, the more similar the two images are. The calculation is performed on various windows of an image. Between windows x

and y, the measurement is calculated by this equation:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.2}$$

where μ_x is the sample mean of window x, μ_y is the sample mean of window y, σ^2 is the variance of their respective windows, σ_{xy} is the covariance of both windows, and c_1 and c_2 are variables that stabilize the division with weak denominators. There's quite a bit to this formula. In essence, it's using the luminance, contrast, and structure of the two windows to calculate the metric. Thankfully, this function is already implemented in skimage so we just use that to do our evaluation.

There are criticisms of both PSNR and SSIM about their reliability as quality measurements. PSNR is considered to be highly influenced by a number of parameters that barely influence visual quality such as brightness, contrast, hue, and saturation [12]. SSIM is more sensitive to spatial shifts and rotations. However, there are good attributes for both parameters. SSIM is more reliable for capturing noise and blurring. PSNR is simple to compute, does fairly well at capturing noise, and has a long history of usage which makes it easy to compare against other works [12] While we are still going to use PSNR and SSIM for these experiments, we'd ideally like an expert to evaluate the quality of our images. We also encourage the exploration of new performance metrics in future works.

The five experiments we perform are sample and batch size tests, noise tests, a Richardson-Lucy comparison, PSF Performance Drop-off tests, and the Real-World qualitative test.

5.1 Sample and Batch Sizes

As mentioned before, we're testing whether or not sampling from the point spread function can give us good performance for a lower computation cost. Because of this, it's important to figure out at what sample size do we reach acceptable results and whether increasing the samples size improves the results. For this test, we use the synthetic microtubules dataset. To get the input, we convolve the ground truth with the larger non-isotropic point spread function. No noise is added to these images. In terms of sample sizes, we start at 1024 samples and go up by powers of two until 16,384 samples.

This section is split off into two sections of testing. We test the change in sample sizes, and we test the change in batch sizes under the assumption that increasing batch size would increase the number of pixels that are updated for each training loop which might lead to better results. For these tests, we fix the number of samples and up the batch sizes starting from 1024 and go up by powers of two until 16,384.

5.2 Noise Tests

The noise tests evaluate how well our algorithm does when images are affected by noise. There are two types of noise distributions that you typically see in SIM images. They are Poisson noise and Gaussian one-shot noise. For theses tests, we use the same synthetic microtubules dataset, but we use the smaller version of the non-isotropic point spread function. We decide to use the smaller version because we want to focus on evaluating the best our algorithm can do on the noise. The best results that we saw early on came from smaller-sized point spread functions. Because of the size, we fix the number of samples at 1024 and the batch size at 8192.

5.3 Richardson-Lucy Comparison

The initial tests evaluated the change in performance of the algorithm when we played with certain hyperparameters. This test focuses on comparing our algorithm results with the results of a defined benchmark. Richardson-Lucy is a very popular and widely-used deconvolution method. However, it suffers from a few common problems such as the lack of stopping criteria, bad performance in the presence of noise, and the introduction of new artifacts that aren't present in the original volume. This test also uses the synthetic microtubules data and the small sized non-isotropic PSF. We do a comparison for images affected with noise and one without.

5.4 Performance Drop-off

Before, we talked about how we used a small non-isotropic point spread function for one subset of tests because the algorithm gave the best results with smaller point spread functions. This subset of tests focuses on quantifying the performance drop-off as we shrink the point spread function size. We go about these tests a bit differently. We still use the synthetic microtubules dataset for the sake of having a ground truth to gather performance metrics with. However, we use multiple point spread functions of varying sizes. The basic type of point spread function follows the Born and Wolf PSF that you can create through the PSF Generator in FIJI. We start from a point spread function of size $63 \times 255 \times 255$ with 63 being the depth of the PSF and the next two parameters being width and height respectively. We then scale down by powers of two except for the very last test where we only adjust the width and height for an even smaller PSF. The input for each test consists of the ground truth convolved with the specific size of point spread function that we are testing. We also fix the sample and batch sizes throughout all of the tests at 1024 and 8192 respectively. Each of the test inputs do not include any noise.

5.5 Real World Test

All the tests before this used a synthetic dataset. For this algorithm to be practical, it has to be performative on real world data. For this, we took a dataset from the EPFL website that featured fluorescence imaging of a C. Elegans embryo. There were three distinct structures of the embryo that were being imaged. Each of the three parts has a point spread function associated with it. We deconvolve each one of the structures using a sample size of 2048 and a batch size of 8192. We also deconvolved one of the structures using a larger sample size of 8192 and batch size of 8192. Since we can't gather PSNR and SSIM values for this dataset, we do a qualitative analysis on the deconvolved images.

Chapter 6

RESULTS

6.1 Samples Sizes and Batch Sizes Results

Table 6.1: Sample Size Performance Results

Sample Size	Batch Size	PSNR	SSIM
1024	8192	22.96	0.72
2048	8192	22.90	0.73
4096	8192	23.18	0.74
8192	8192	23.44	0.73
16,384	8192	23.60	0.74

In Table 6.1, we can see that as we increase the number of PSF samples used in the deconvolution process, our PSNR scores gradually improve. However, the improvement does not scale along with the growth of the sample sizes. From doubling the sample size from 8192 samples to 16,384, the PSNR only improves by 0.16. On average, we get a 0.16 improvement to the PSNR every time we double the sample size from 1024 to 16,384. As for SSIM, the scores remain fairly consistent throughout the sample size changes, hovering right around 0.73-0.74.

Table 6.2: Batch Size Performance Results

Sample Size	Batch Size	PSNR	SSIM
2048	1024	22.87	0.73
2048	2048	22.74	0.74
2048	4096	23.44	0.73
2048	8192	22.90	0.73
2048	16,384	22.89	0.74

Changing the batch size does not have the same effects as changing the sample size. Table 6.2 shows us that there doesn't appear to be any consistent relationship between increasing the batch size and our PSNR value. For these tests, the PSNR tops out at 23.44 at a batch size of 4096 but quickly declines for the two increased batch sizes after. Just like the sample size tests, batch sizes also appear to have no affect on the SSIM score with it remaining in the same range of 0.73-0.74.

6.2 Noise Tests

Table 6.3: Noise Performance Results

Gaussian Noise (Stdev)	Poisson Noise	PSNR	SSIM
1600	400	27.40	0.83
3200	800	26.84	0.83
6400	1600	26.39	0.83
12,800	3200	25.54	0.81

This set of tests analyzes the performance of the PSF sampling method under noise conditions. Each volume has a mixture of Gaussian noise and Poisson noise. The results can be seen in Table 6.3. Not surprisingly as the value of noise increases in the volumes, our PSNR values slowly degrade as well. SSIM is a bit more resilient to the increases in noise but still shows signs of degrading in the last test. Doubling the noise at each level leads to an average decrease of 0.62 in the PSNR value.

6.3 Richardson-Lucy Comparison

Table 6.4: RL PSNR Results on Noise Simulations

Noise (G, P)	RL 20	RL 50	RL 100	RL 150	RL 200	PSF-S
0, 0	33.95	34.85	35.59	35.96	36.10	34.78
100, 25	28.15	28.22	28.25	28.26	28.26	28.01
200, 50	28.15	28.23	28.26	28.27	28.27	28.05
400, 100	28.16	28.24	25.71	N/A	N/A	27.89
800, 200	28.16	25.69	N/A	N/A	N/A	27.70

Table 6.4 shows the PSNR results of running the Richardson-Lucy deconvolution algorithm on the synthetic microtubules dataset. The number next to the ”RL” denotes the number of iterations performed for the deconvolution. PSF-S denotes our method, short for ”PSF Sampling”. The noise column shows the amount of noise present in each of the volumes used for the test. For images with no noise, RL reconstructs the volume very well and even outperforms our sampling method at the smallest number of iterations. However, the performance of RL quickly declines as the amount of noise increases.

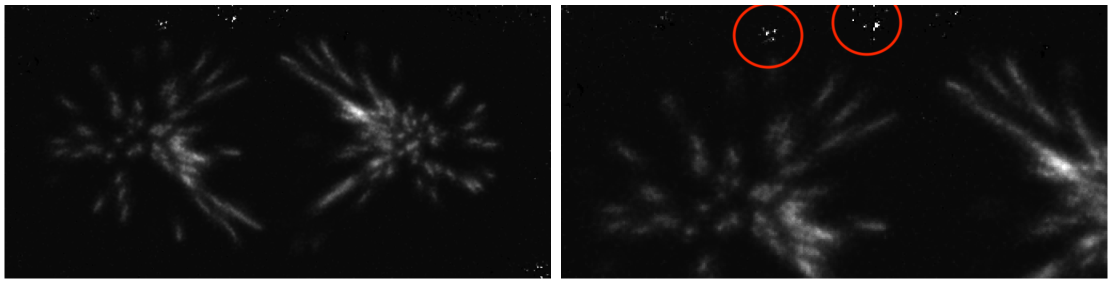


Figure 6.1: Amplified noise in Richardson-Lucy deconvolved images.

Because of the way the RL algorithm works, the noise in the volume has the potential to be amplified after a number of iterations. Figure 6.1 shows the result of the last test using RL with 50 iterations. It highlights the amplified noise that did not appear in the original test volume. The ”N/A”s that appear in the table are due to those volumes becoming extremely corrupted by noise after running that many iterations. One important thing to note here is the difference between the amount of noise used for this set of tests and the amount of noise used for the tests in Table 6.3. Even though the PSNR values steadily decrease, our algorithm remains somewhat stable and doesn't allow the noise to explode in the volume.

Table 6.5: RL SSIM Results on Noise Simulations

Noise (G, P)	RL 20	RL 50	RL 100	RL 150	RL 200	PSF-S
0, 0	0.98	0.99	0.99	0.99	0.99	0.92
100, 25	0.90	0.90	0.90	0.90	0.90	0.83
200, 50	0.90	0.90	0.90	0.90	0.90	0.83
400, 100	0.90	0.90	0.88	N/A	N/A	0.83
800, 200	0.90	0.88	N/A	N/A	N/A	0.83

Table 6.5 shows the SSIM results for the same set of tests as the PSNR table. It basically tells us the same story. RL does really well under perfect conditions but shows reduced performance as noise increases. Like the SSIM tests before, our algorithm remains consistent as the level of noise increases.

Below in Figure 6.2, we can see slight differences in the results of the two methods. With our sampling method, we appear to have a brighter signal at the cost of having more pronounced noise in the image. The noise in the RL image isn't as noticeable, but the signal also looks a bit dimmer.

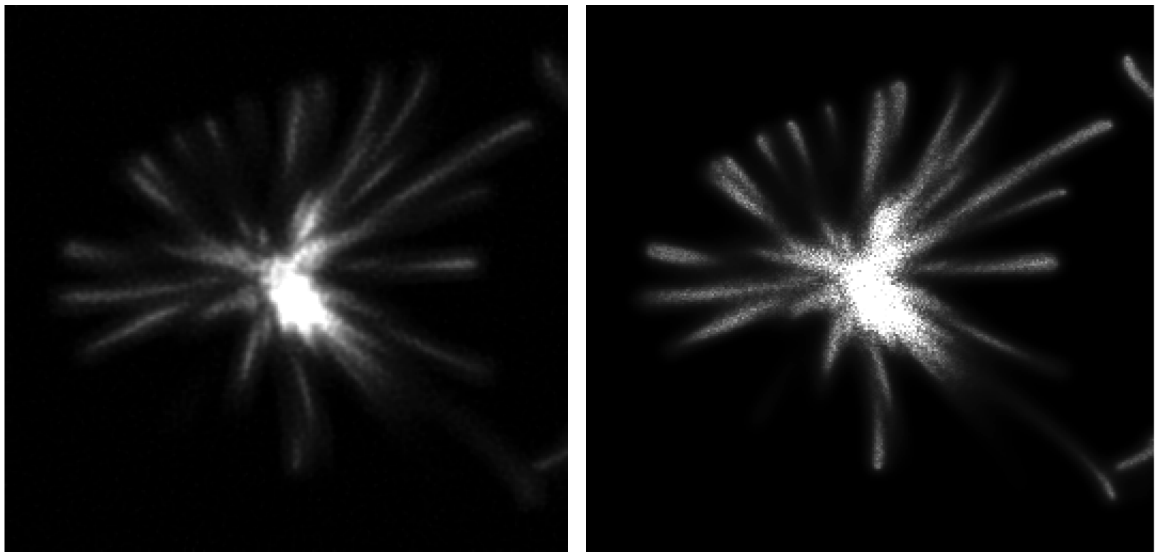


Figure 6.2: RL method (a) vs PSF sampling (b) on noise afflicted images.

6.4 PSF Performance Drop-off

Table 6.6: PSF Performance Drop-off Results

PSF Size (z, y, x)	PSNR	SSIM
7, 15, 15	35.53	0.92
7, 31, 31	34.87	0.92
15, 63, 63	31.72	0.89
31, 127, 127	26.66	0.82
63, 255, 255	21.46	0.69

In Table 6.6, we can see the results from the PSF tests. The table shows us that smaller PSFs tend to score very high PSNR and SSIM values. However, as we increase the size of the PSF which increases the size of our sample space, the performance of our method suffers. The smallest PSF size scored a 35.53 PSNR and a 0.92 SSIM while the biggest PSF dropped those scores to 21.46 and 0.69 respectively.

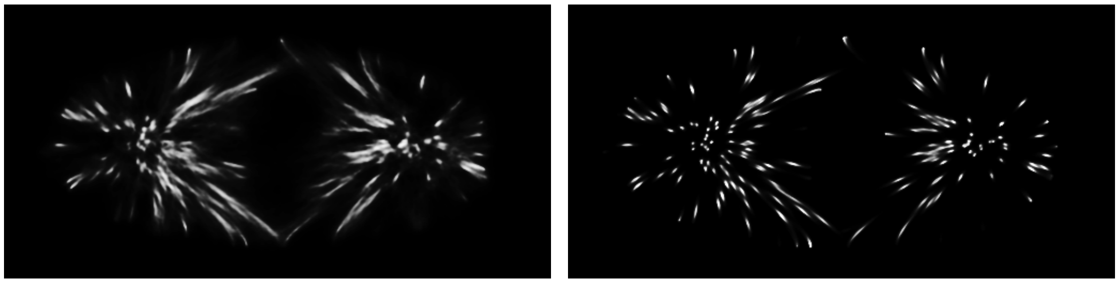


Figure 6.3: Comparison of deconvolved microtubules using the biggest PSF (a) and the smallest PSF (b).

The difference in performance is also very noticeable in the images themselves. Figure 6.3 highlights the difference in clarity between the volume deconvolved with the biggest PSF and the one that used the smallest. The microtubules in the left image still have a large blur effect whereas the light in image on the right is much more concentrated in specific areas.

6.5 Real-World Qualitative Results

Lastly, we have the results of the real-world dataset. Since we don't have a ground truth for these images, we resort to doing a qualitative analysis. To help, some of the images are converted to a heat color scheme to better show the intensity of the signals.

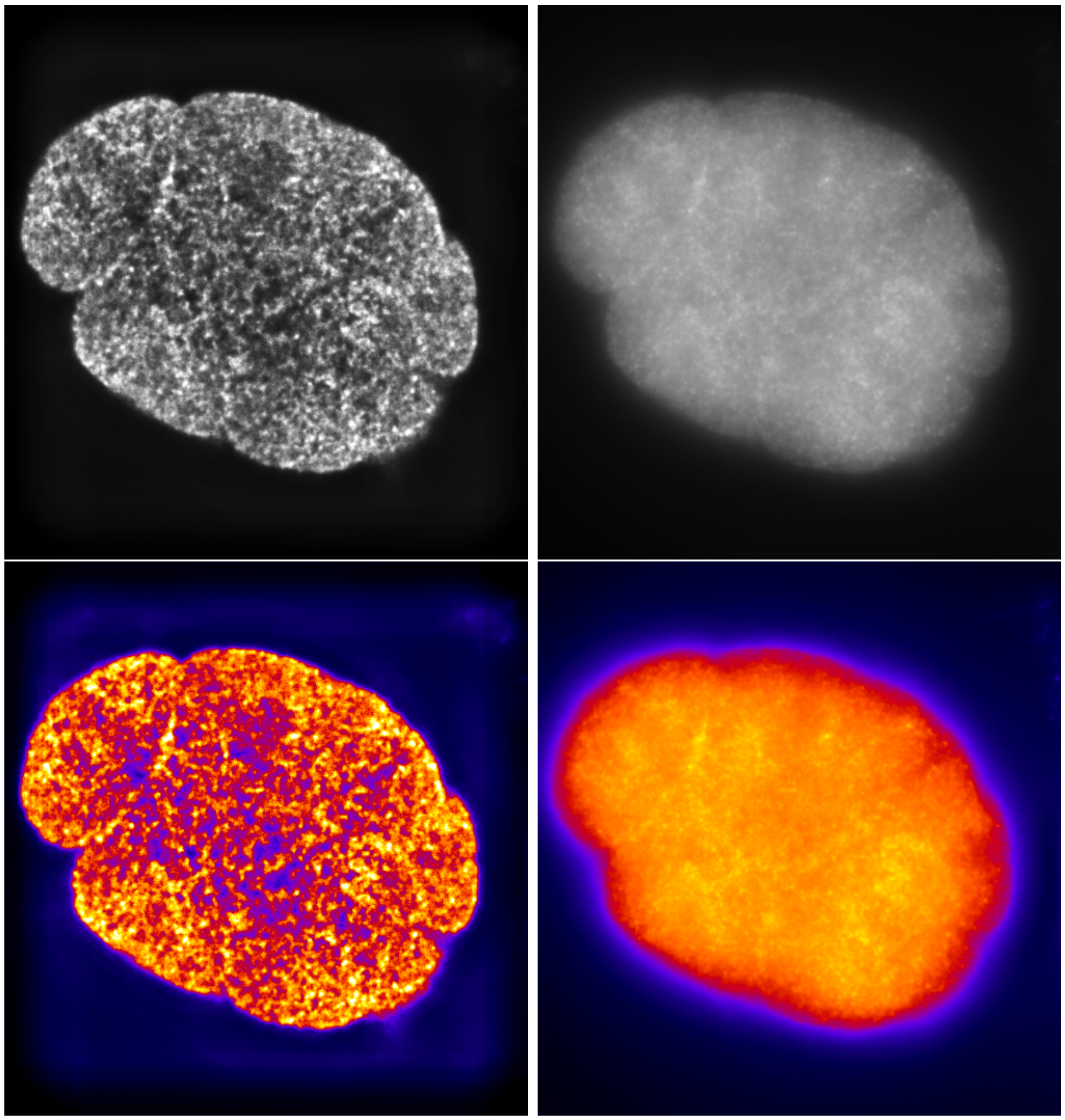


Figure 6.4: CY3 grayscale and intensity images of both deconvolved (left) and convolved (right) volumes.

Figure 6.4 shows the original and deconvolved volumes of the CY3 channel of the C. Elegans embryo. This test used 8192 samples and a batch size of 8192. At first glance, our method significantly clears up a lot of the blur. In the heat-colored deconvolved image, you can easily see more concentrated areas of light as compared to its convolved counterpart. This test featured the largest sample size used to deconvolve any of the embryo volumes. To see if the number of samples made a significant difference in this context, we deconvolved the same CY3 volume with 2048 samples.

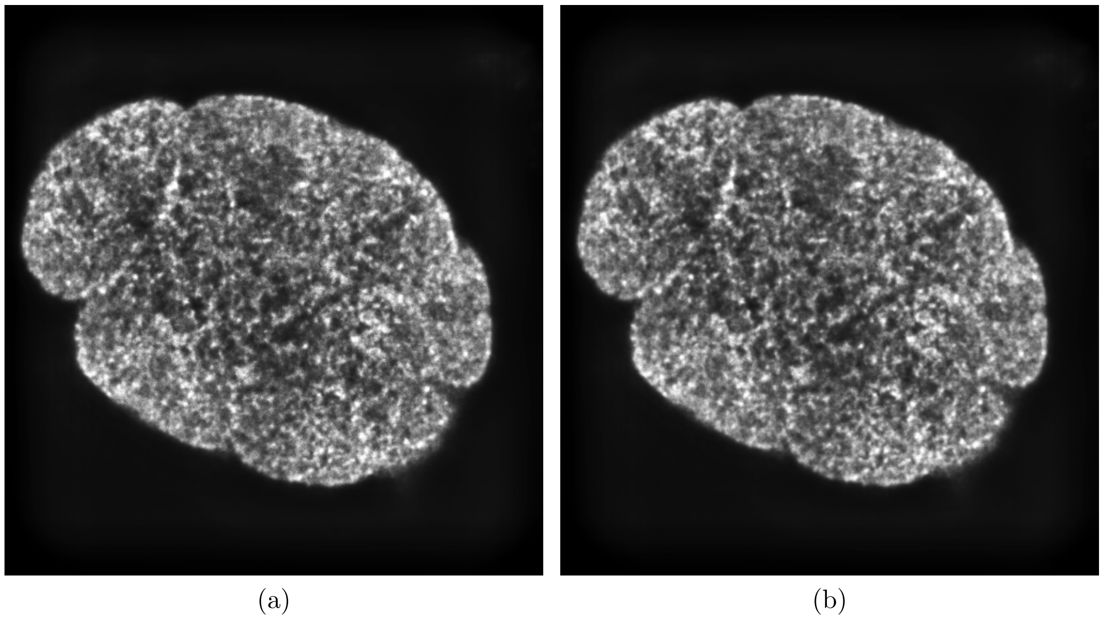


Figure 6.5: Deconvolved CY3 images with 8192 samples (a) and 2048 samples (b).

Figure 6.5 shows the results of both the 8192 sample and 2048 sample deconvolutions together side by side. Even after inspecting the heat-colored versions of these images, we could not discern any significant difference between the results of the two sample sizes.

The next channel we show here is the DAPI channel, or the chromosomes in the nuclei of the embyro.

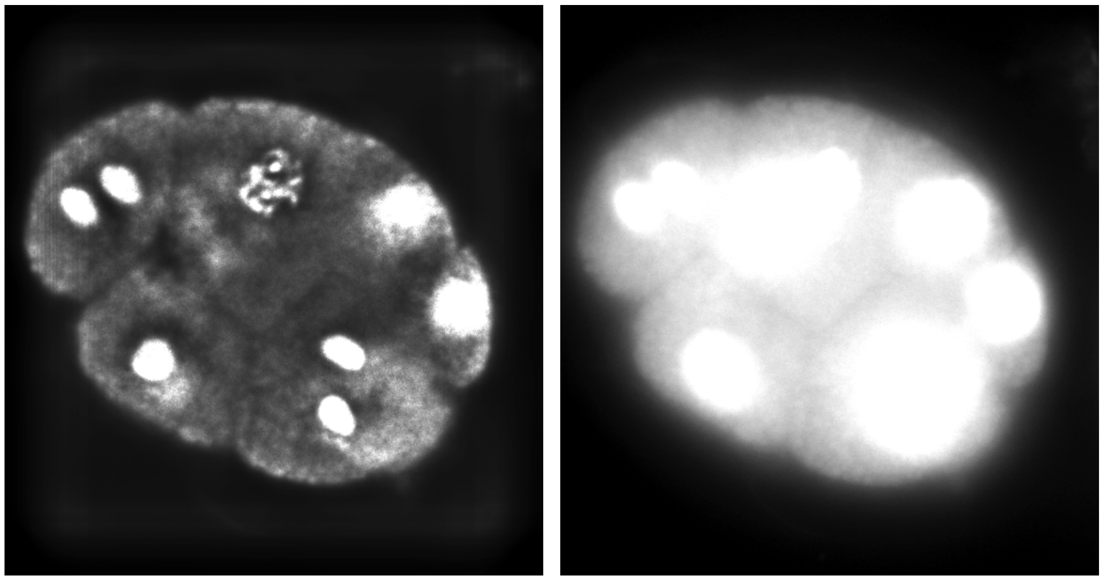

(a) (b)

Figure 6.6: Deconvoled DAPI channel (a) vs. convolved DAPI channel (b).

The convolved image in Figure 6.6 features signals that are too bright to see. The locations of the nuclei are very hard to pinpoint exactly. The result of our algorithm using 2048 samples drastically improves the blurring around the locations of interest which really helps with seeing where they are. The deconvolved images also highlight an interesting behavior of the nuclei. Only one nuclei in the embryo is allowed to split at a time. The nuclei in the top middle section of the embryo is the one splitting at this point in time. We know it's splitting because the chromosomes are more visible than the other nuclei. While the chromosomes are still present in the other nuclei, they are very difficult to see because of how tightly packed together they are. We take a closer look at the splitting nuclei in the next figure.

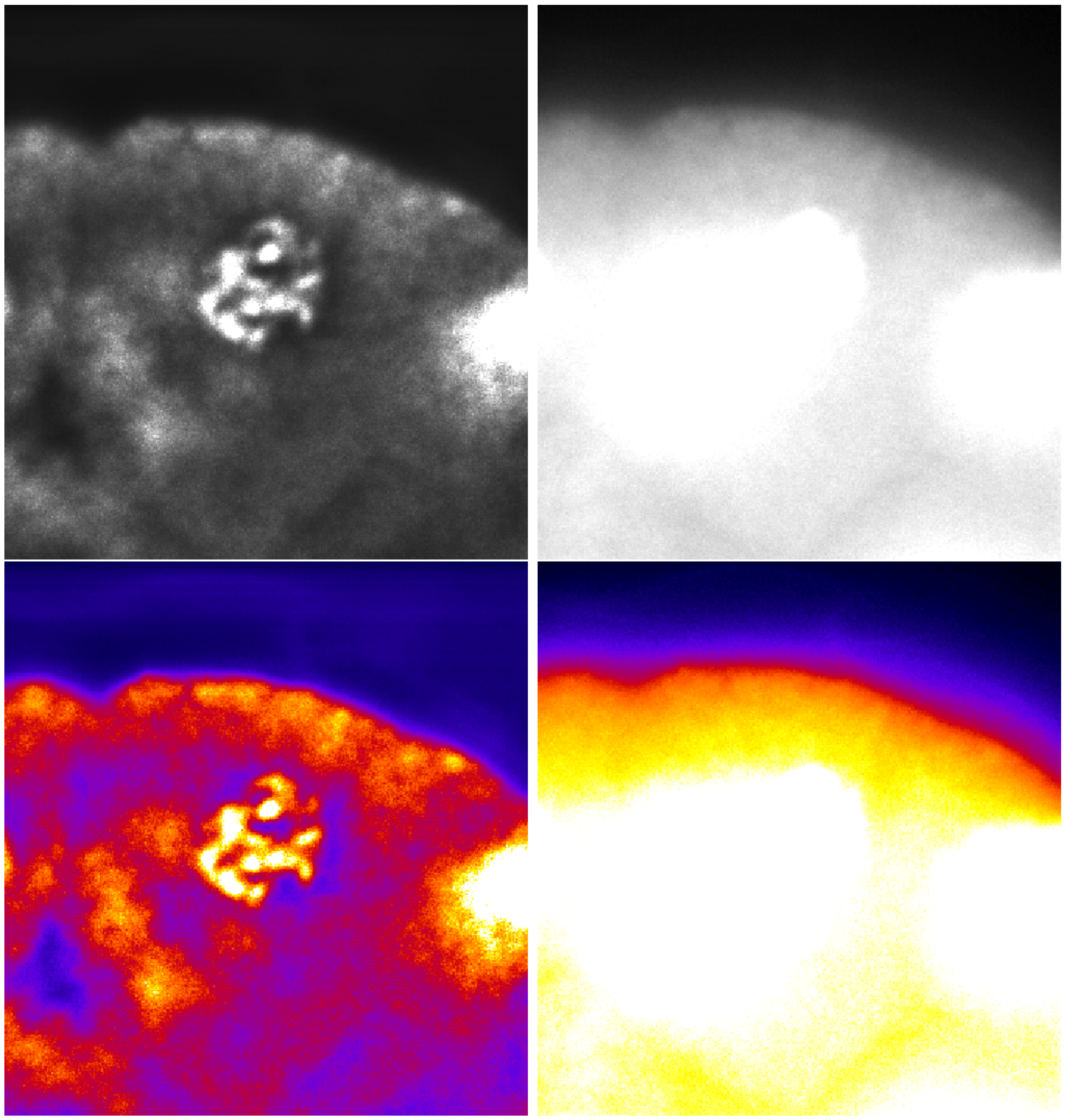


Figure 6.7: A slightly resolved nuclei in the DAPI channel.

Figure 6.7 highlights this small nuclei in the upper-middle part of the embryo. As mentioned before, there's a significant improvement in these images from the original to the deconvolved results. Because this is the only nuclei splitting at this point in time, we are able to barely discern what are supposed to be chromosomes in the structure. It is still very difficult to identify any of the finer parts of the chromosome. The heat-colored images in the figure serve to further show the difference in light concentration. Below we also show a close up of a nuclei with tightly packed chromosomes in Figure 6.8.

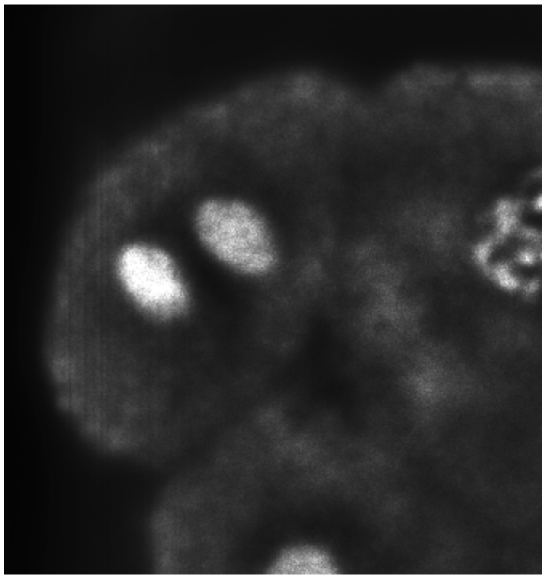


Figure 6.8: Tightly packed nuclei in deconvolved DAPI.

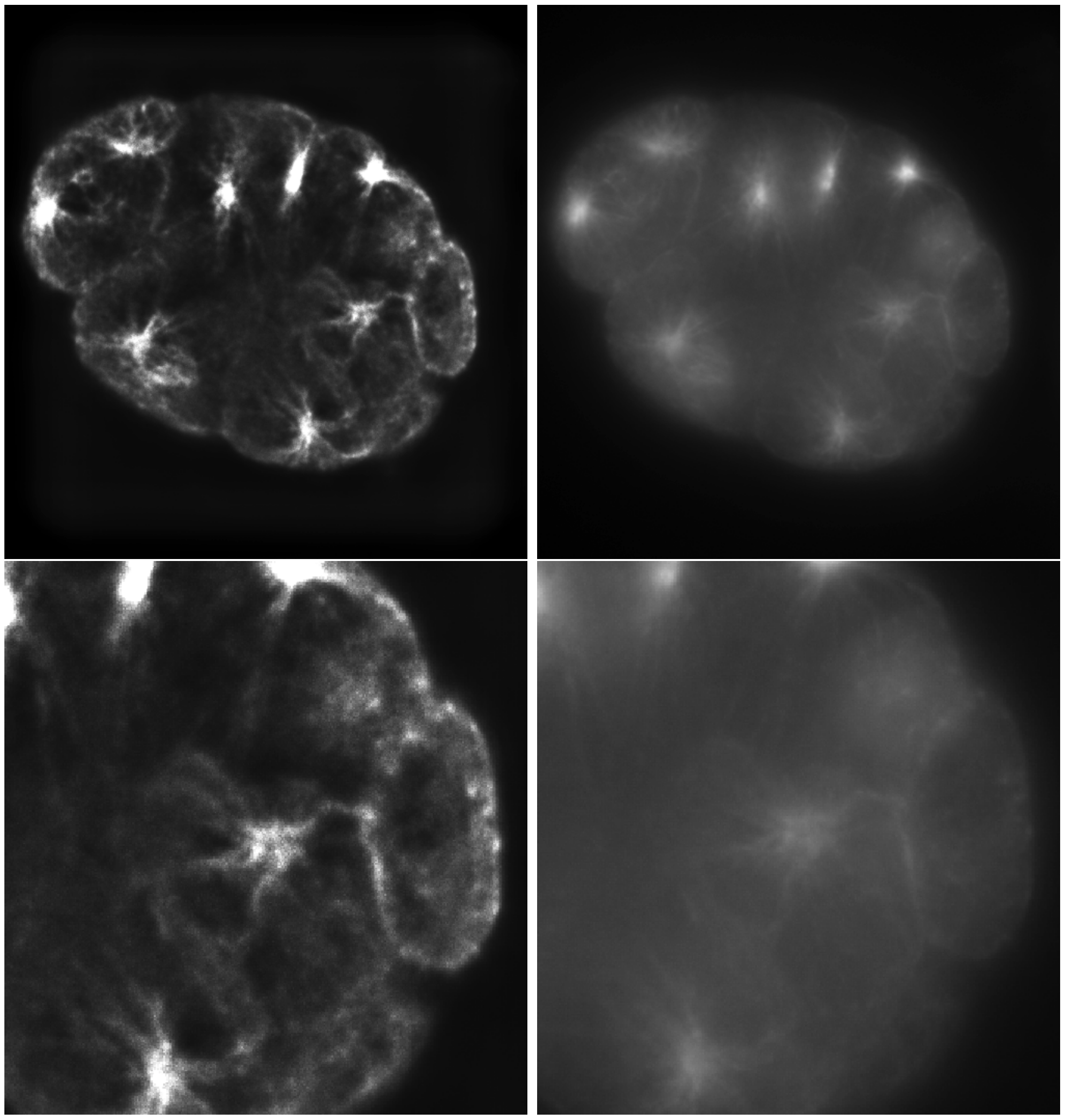


Figure 6.9: Deconvolved FITC channel whole view (top) and zoomed view (bottom).

Lastly, we have the results of deconvolving the FITC channel of the embryo in Figure 6.9. The top set of images features a whole view of the microtubules. The bottom set of images takes a closer look at one section of the microtubules. Like the images before, our method appears to do very well at reducing blur throughout the entirety of the image. However, the method does struggle when trying to refine structures. The light doesn't appear to be as concentrated as it should be within the structures.

For example in the top set of images, the light does get concentrated in the area that the microtubule appears in, but that microtubule lacks the finer light concentration needed to view the smaller details. In the bottom set of images in Figure 6.9, we can see a set of microtubules that becomes clearer after the deconvolution.

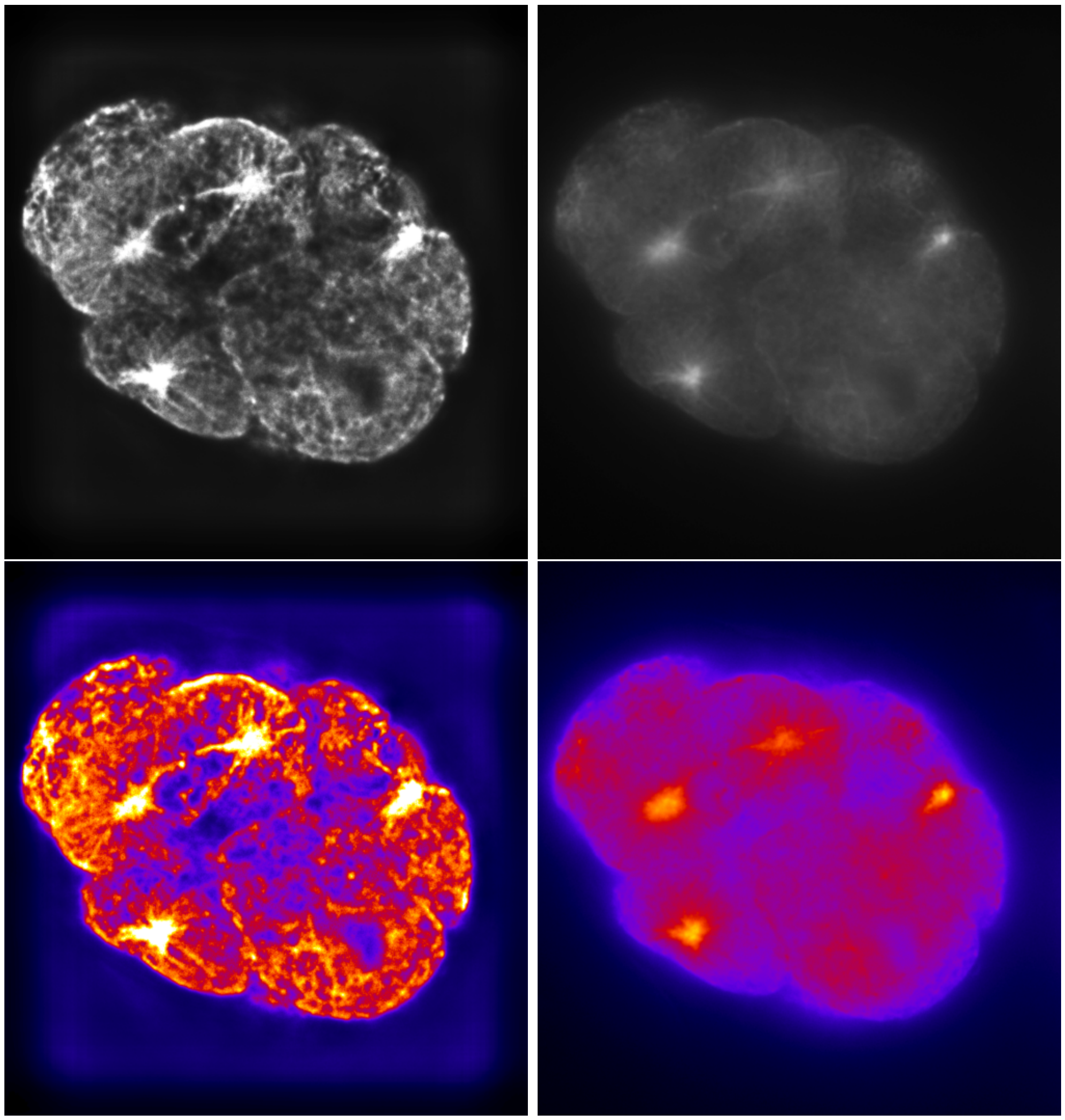


Figure 6.10: Slide 71 of the FITC volume.

These next set of images in Figure 6.10 show the deconvolved and convolved versions of a different slice of the FITC volume. These images are interesting because they display an overall brighter signal after the deconvolution, similar to what we saw in Figure 6.2b. This difference is made clearer with the intensity images on the bottom.

Chapter 7

DISCUSSION

7.1 Performance Limitations

To deconvolve images using this method, there are a few factors that play into the performance of the algorithm, at least for our implementation. These factors mostly relate to the PSF sampling. The first and most obvious factor is the sample size. As the number of samples increases, so does the computation time. The more samples there are, the more points a single pixel gets convolved with. The second, and probably less obvious factor, is the size of the point spread function. The way TensorFlow's $tf.random.categorical$ method is implemented plays a role in this. The function tries to create a tensor of shape $1 \times \textit{sample size} \times \textit{sample space}$ on the GPU. With a flattened PSF serving as our sample space and a decent sample size, that tensor can potentially be quite large and will throw an error if the GPU does not have enough memory. Our largest test, the first CY3 test, used a sample size of 8192 and had a PSF of size $104 \times 672 \times 712$ which, when flattened, comes out to a dimension of 49,760,256. The test took roughly 6 days to finish. Lastly, the same thing is true for the batch size. As the batch size increases, the computation time also increases although not as drastically as the PSF size. These are just things to be aware of should one try to test this method.

7.2 Future Work

One potential idea for future work in this project is to experiment with sampling without replacement. Because we sample with replacement in our work, it is possible that some areas of the volume are over-contributing to the fluorescence density for a certain pixel. Technically, it's also another way to approximate the convolution. If we treat the values of the point spread function as coefficients, we could sample locations without replacement, weigh them with the coefficients, and average the samples with the sum of the coefficients. This might give a closer estimate than our sampling method and would be very interesting to explore in future work.

Another area of work is the representation. In the early versions of this project, we experimented with using a NeRF representation of the volume. Our initial experience didn't work out so great as the training process was super slow, and the loss didn't converge very well. Despite this, we still believe there's a lot of potential in either continuing with NeRF or making use of other representations.

In terms of performance improvement, implementing the multiresolution hash table that was mentioned earlier in the related works might be worthwhile. Considering the time it takes for the algorithm to run, any optimization to the method would be very beneficial for testing. Another thing to look into would be potentially distributing the work across multiple GPUs. There's already been work done parallelizing iterative deconvolution methods for large-scale fluorescence microscopy data [20]. If distributing the work across multiple devices is possible with our method, it would be incredibly helpful.

Chapter 8

CONCLUSION

In this project, we experimented with a slight variation of deconvolving fluorescence microscopy images through sampling the point spread function. Our method was tested under multiple conditions using a synthetic dataset for ground-truth values and a real-world dataset featuring a C. Elegans embryo. To compare PSNR and SSIM results, we used the original Richardson-Lucy deconvolution algorithm as a baseline. Our model uses an explicit representation of the volume and a 3D grid of offsets for the point spread function to be able to reconstruct the original, unblurred volume. In our results, increasing the sample size of the point spread function tends to increase the PSNR value while having very little effect on the SSIM. In the presence of noise, our method shows performance declines in both PSNR and SSIM, however, the volumes don't become corrupted by noise-amplification like Richardson-Lucy. In the baseline comparison, Richardson-Lucy outperforms our method under the condition that very little noise is present in the volume. Past a certain point of noise, our sampling method appears to be the better choice. While this algorithm also shows really good improvements to real-world fluorescence images, it does exhibit performance drops as the size of the point spread function increases. Overall, this is a novel take on fluorescence deconvolution that presents a stable alternative to the base Richardson-Lucy method for deconvolving noisy fluorescence microscopy images.

BIBLIOGRAPHY

[1] 3D Deconvolution Microscopy. http://bigwww.epfl.ch/deconvolution/bio/.

[2] Cal Poly Github. `http://www.github.com/CalPoly`.

[3] Digital Image Processing - Algorithms for Deconvolution Microscopy | Olympus LS. https://www.olympus-lifescience.com/en/microscope-resource/primer/digitalimaging/deconvolution/deconalgorithms/.

[4] Digital Image Processing - Artifacts and Aberrations in Deconvolution Analysis | Olympus LS. https://www.olympus-lifescience.com/en/microscope-resource/primer/digitalimaging/deconvolution/deconartifacts/.

[5] Digital Image Processing - Introduction to Deconvolution | Olympus LS. https://www.olympus-lifescience.com/en/microscope-resource/primer/digitalimaging/deconvolution/deconintro/.

[6] EPFL | Biomedical Imaging Group | Laboratory. http://bigwww.epfl.ch/.

[7] Fluorescent Microscopy. https://serc.carleton.edu/microbelife/research_methods/microscopy/.

[8] Introduction to Fluorescence Microscopy | Nikon's MicroscopyU. https://www.microscopyu.com/techniques/fluorescence/introduction-to-fluorescence-microscopy.

[9] Neural Radiance Field (NeRF): A Gentle Introduction. https://datagen.tech/guides/synthetic-data/neural-radiance-field-nerf/.

[10] NumPy. https://numpy.org/.

[11] Pillow. https://pillow.readthedocs.io/en/stable/index.html.

[12] PSNR and SSIM: application areas and criticism.

[13] TensorFlow. https://www.tensorflow.org/.

[14] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, and P. P. Srinivasan. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5835–5844, Montreal, QC, Canada, Oct. 2021. IEEE.

[15] N. Chakrova, B. Rieger, and S. Stallinga. Deconvolution methods for structured illumination microscopy. *Journal of the Optical Society of America A*, 33(7):B12, July 2016.

[16] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su. TensoRF: Tensorial Radiance Fields, Nov. 2022. arXiv:2203.09517 [cs].

[17] Y. Chen, M. Chen, L. Zhu, J. Y. Wu, S. Du, and Y. Li. Measure and model a 3-D space-variant PSF for fluorescence microscopy image deblurring. *Optics Express*, 26(11):14375, May 2018.

[18] P. P. Laissue, R. A. Alghamdi, P. Tomancak, E. G. Reynaud, and H. Shroff. Assessing phototoxicity in live fluorescence imaging. *Nature Methods*, 14(7):657–661, July 2017.

[19] S. Lee, S. Han, P. Salama, K. W. Dunn, and E. J. Delp. Three Dimensional Blind Image Deconvolution for Fluorescence Microscopy using Generative Adversarial Networks. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, pages 538–542, Venice, Italy, Apr. 2019. IEEE.

[20] G. Lu, M. A. Baertsch, J. W. Hickey, Y. Goltsev, A. J. Rech, L. Mani, E. Forgó, C. Kong, S. Jiang, G. P. Nolan, and E. L. Rosenthal. A real-time GPU-accelerated parallelized image processor for large-scale multiplexed fluorescence microscopy data. *Frontiers in Immunology*, 13:981825, Sept. 2022.

[21] L. Ma, X. Li, J. Liao, Q. Zhang, X. Wang, J. Wang, and P. V. Sander. Deblur-NeRF: Neural Radiance Fields from Blurry Images, Mar. 2022. arXiv:2111.14292 [cs].

[22] M. Makarkin and D. Bratashov. State-of-the-Art Approaches for Image Deconvolution Problems, including Modern Deep Learning Architectures. *Micromachines*, 12(12):1558, Dec. 2021.

[23] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis.

[24] T. Müller, A. Evans, C. Schied, and A. Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics*, 41(4):1–15, July 2022.

[25] A. Reed, T. Blanford, D. C. Brown, and S. Jayasuriya. Implicit Neural Representations for Deconvolving SAS Images. In *OCEANS 2021: San Diego – Porto*, pages 1–7, San Diego, CA, USA, Sept. 2021. IEEE.

[26] D. Sage, L. Donati, F. Soulez, D. Fortun, G. Schmit, A. Seitz, R. Guiet, C. Vonesch, and M. Unser. DeconvolutionLab2: An open-source software for deconvolution microscopy. *Methods*, 115:28–41, Feb. 2017.

[27] P. Sarder and A. Nehorai. Deconvolution methods for 3-D fluorescence microscopy images. *IEEE Signal Processing Magazine*, 23(3):32–45, May 2006.

[28] M. Sticker, R. Elsässer, M. Neumann, and H. Wolff. How to Get Better Fluorescence Images with Your Widefield Microscope: A Methodology Review. *Microscopy Today*, 28(6):36–43, Nov. 2020.

[29] M. Weigert, U. Schmidt, T. Boothe, A. Muller, A. Dibrov, A. Jain, B. Wilhelm, D. Schmidt, C. Broaddus, S. Culley, M. Rocha-Martins, F. Segovia-Miranda, C. Norden, R. Henriques, M. Solimena, J. Rink, P. Tomancak, L. Royer, F. Jug, and E. W. Myers. Content-Aware Image Restoration: Pushing the Limits of Fluorescence Microscopy.

[30] W. Zhao, S. Zhao, L. Li, X. Huang, S. Xing, Y. Zhang, G. Qiu, Z. Han, Y. Shang, D.-e. Sun, C. Shan, R. Wu, L. Gu, S. Zhang, R. Chen, J. Xiao, Y. Mo, J. Wang, W. Ji, X. Chen, B. Ding, Y. Liu, H. Mao, B.-L. Song, J. Tan, J. Liu, H. Li, and L. Chen. Sparse deconvolution improves the resolution of live-cell super-resolution fluorescence microscopy. *Nature Biotechnology*, 40(4):606–617, Apr. 2022.

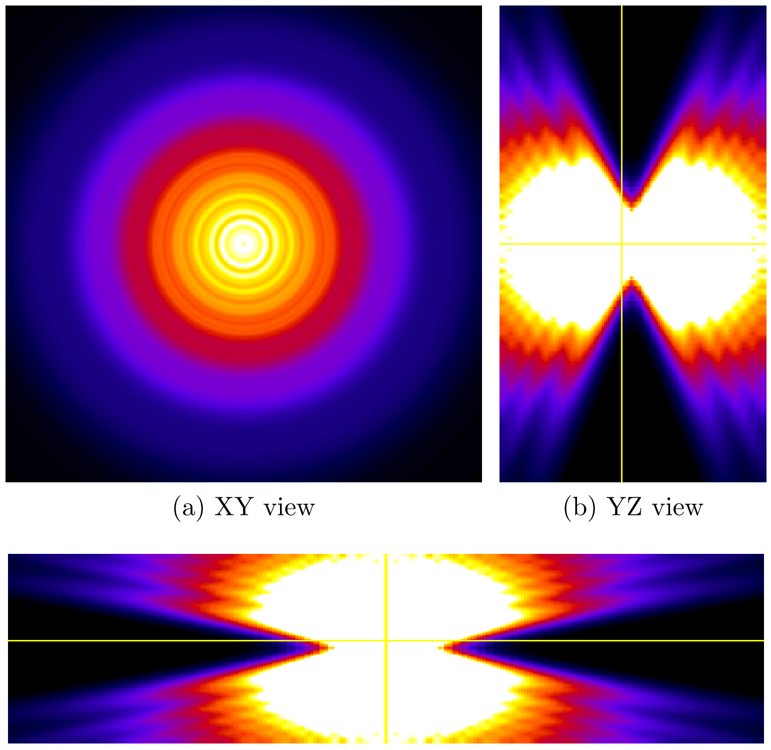
[31] E. D. Zhong, T. Bepler, J. H. Davis, and B. Berger. Reconstructing continuous distributions of 3D protein structure from cryo-EM images, Feb. 2020. arXiv:1909.05215 [cs, eess, q-bio, stat].

APPENDICES

Appendix A

POINT SPREAD FUNCTIONS

A.1 Non-Isotropic PSFs


(a) XY view (b) YZ view

(c) XZ view

Figure A.1: Large Non-Isotropic PSF

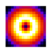
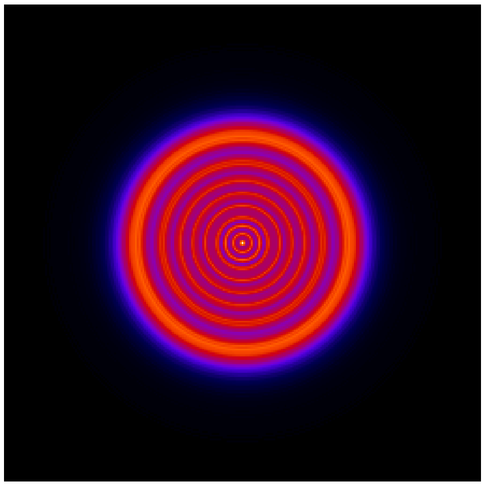
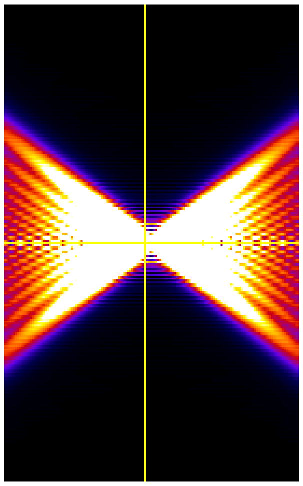


Figure A.2: Small Non-Isotropic PSF (7 x 15 x 15)

A.2 Born and Wolf PSFs


(a) XY view


(b) YZ view

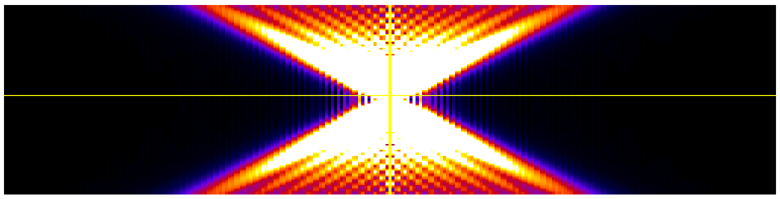
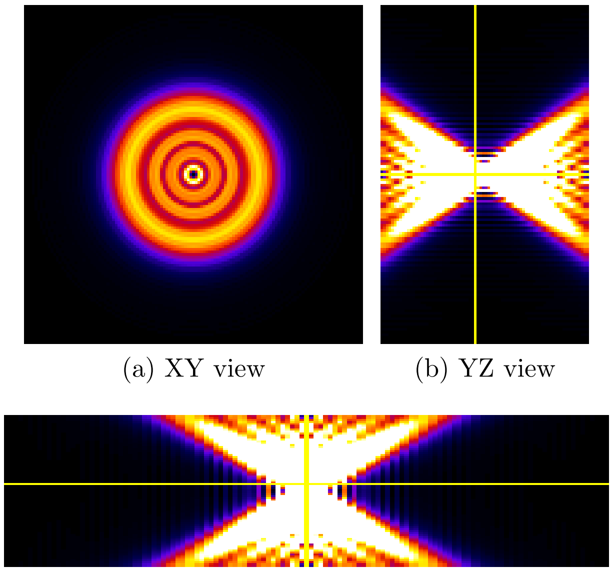

(c) XZ view

Figure A.3: Largest B&W PSF (63 x 255 x 255)


(a) XY view (b) YZ view

(c) XZ view

Figure A.4: B&W PSF (31 x 127 x 127)

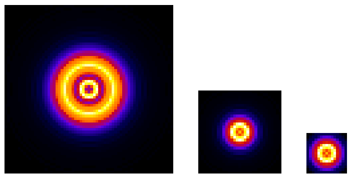
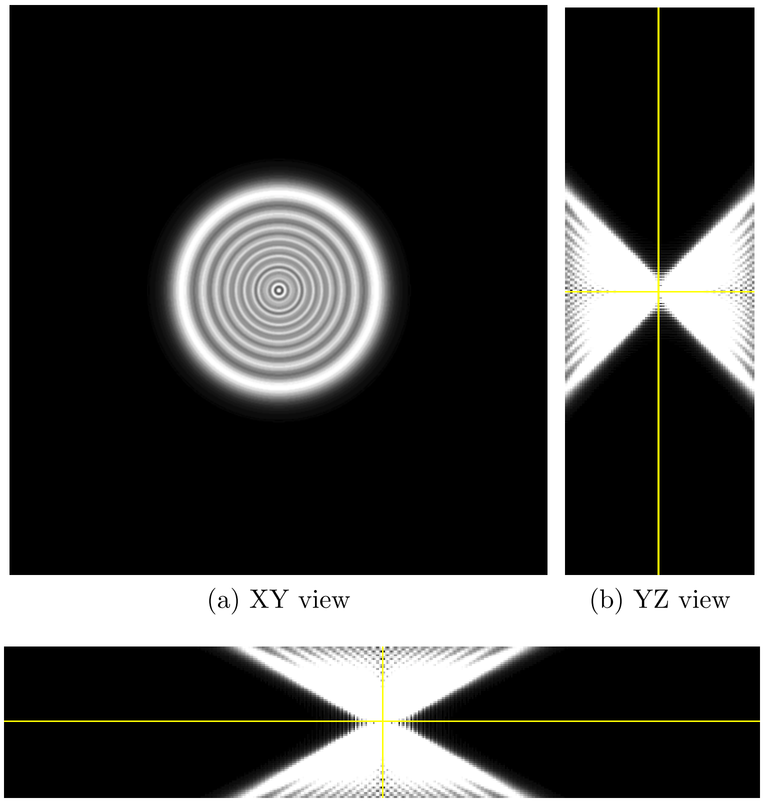

(a) (b) (c)

Figure A.5: The three smallest B&W PSFs where (a) is size (15 x 63 x 63), (b) is size (7 x 31 x 31), and (c) is size (7 x 15 x 15)

A.3 C. Elegans Embryo PSFs

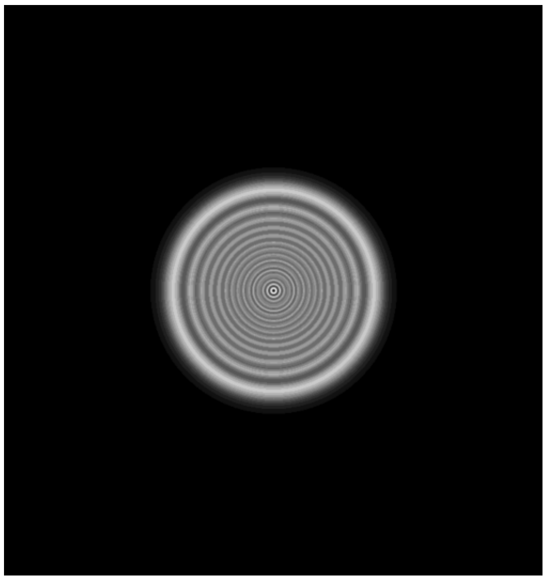


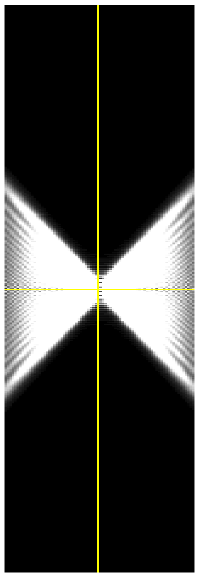
(a) XY view

(b) YZ view

(c) XZ view

Figure A.6: PSF for the CY3 Channel of the C. Elegans Embryo


(a) XY view


(b) YZ view

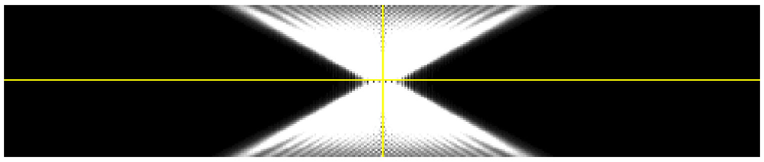
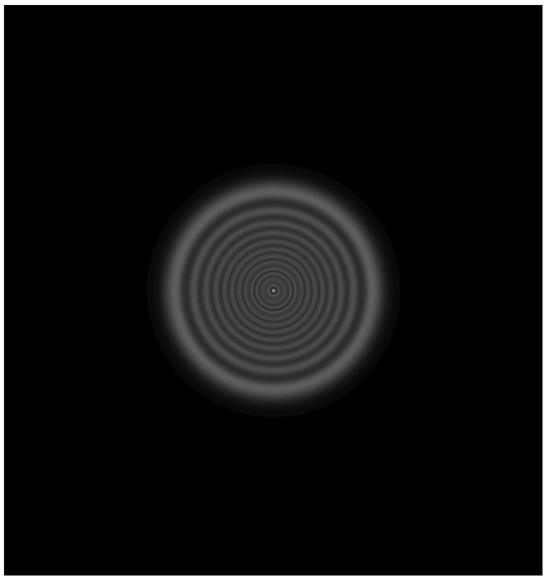
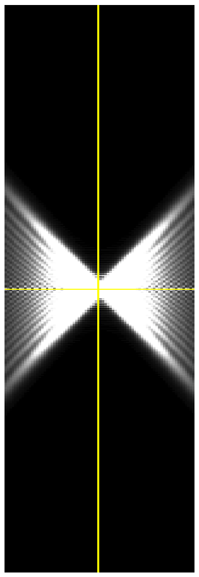

(c) XZ view

Figure A.7: PSF for the DAPI Channel of the C. Elegans Embryo


(a) XY view


(b) YZ view

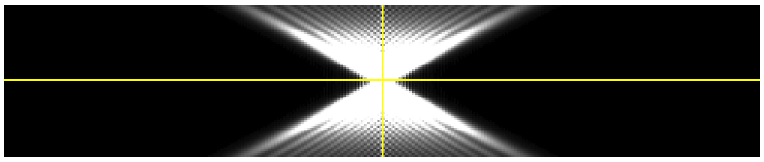

(c) XZ view

Figure A.8: PSF for the FITC Channel of the C. Elegans Embryo