Volumetric Reconstruction of Cellular Organelles using Z-splitter Microscopy

Student (Author)
Han Changhee
rmkroar97@yonsei.ac.kr

Advisory Graduate
Ko Kwanhwi
ko.kwanhwi@gmail.com

Advisory Professor
Kim Donghyun
kimd@yonsei.ac.kr

Department of Electrical and Electronic Engineering
Yonsei University, Seoul 03722, Korea

Abstract

The three-dimensional visualization of cellular organelles is essential for scientific insights and understanding rapid biological processes. In this study, images of mitochondria and lysosomes acquired using z-splitter microscopy are deconvolved and reconstructed through deep learning–based interpolation methods. This study aims to show the feasibility of these approaches in generating volumetric representations that include randomly designed organelle-like artifacts, and to implement a prototype visualization system.

1. Introduction

In the modern field of biological sciences, precise observation and visualization at the cellular level are essential not only for drawing accurate interpretations of research findings, but also plays an important role in inspiring new scientific ideas. Although two-dimensional imaging using optical microscopy has long been a standard method to observe cells, it provides only flat, single-plane information, which limits the ability to fully interpret the three-dimensional structures and dynamics of cellular processes.

The most straightforward but also the slowest method for obtaining volumetric cellular images is to capture multiple two-dimensional images while translating the camera sensor. Image stacks can be acquired more rapidly using fast z-scanners [5] or light-sheet geometries [2], but these techniques are still too slow to capture biological processes such as dielectrophoresis, which can occur on millisecond time scales.

One promising approach for gathering instantaneous volumetric data is z-splitter imaging [12], which has been cited and further developed in various medical and biological studies [9, 14]. This strategy employs a z-splitter prism to split the microscope's detection path into multiple channels of increasing optical path lengths, allowing each channel to capture an image from a different depth within the sample. Additional clarity and improved image quality can be achieved through some appropriate deconvolution methods commonly applied in confocal microscopy [7].

Since z-splitter imaging captures only a limited number of z-planes from the sample, the full volumetric shape must be reconstructed through interpolation along the z-axis. While this can be accomplished using traditional methods such as B-spline interpolation [13], the advent of neural networks has enabled more elaborate deep learning–based interpolation for z-axis upsampling, including 3D U-Net [1] and deep-Z [10].

In this research, the z-stack images of cellular organelles (mitochondria and lysosomes) are assumed as input data. These images are subjected to deconvolution and subsequently reconstructed using deep learning–based interpolation to generate complete volumetric visualizations. We will find out the accuracy and efficiency of this pipeline, and its applicability to real biological imaging will be tested using nanobead samples.

2. Method

2.1. Z-Splitter Imaging

The raw six-plane image stacks of the nanobead samples are gratefully provided by the advisory graduate of this research. The data are in TIFF format with 100 frames each composed of six sections measuring $500\text{nm} \times 500\text{nm}$, captured 50 frames per second, as shown in Figure 1. The nanobead samples are imaged using fluorescence excitation with a 480nm laser and are based on *FluoSpheres*^{TM} *Carboxylate-Modified Microspheres (F8803)*, a 0.1μm-diameter fluorescent nanobead product manufactured by Invitrogen$^{\text{TM}}$. The structure of z-splitter microscope is based on the schematics proposed by xiao et al. [12]

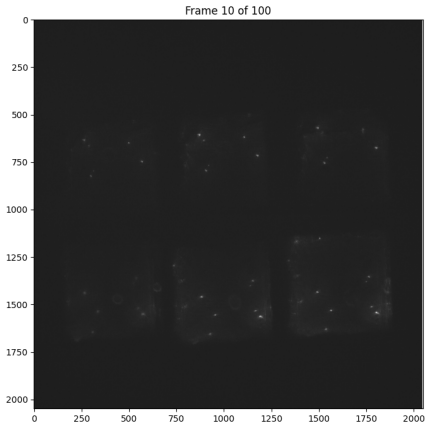


Figure 1. A single frame of nanobead samples obtained with z-splitter microscopy. Glowing dots on the image show the location of fluorescent nanobead particles, and circular, faint imagery are recognizable common in all z-stacks. There exists noise and out-of-focus background all over the images to be suppressed.

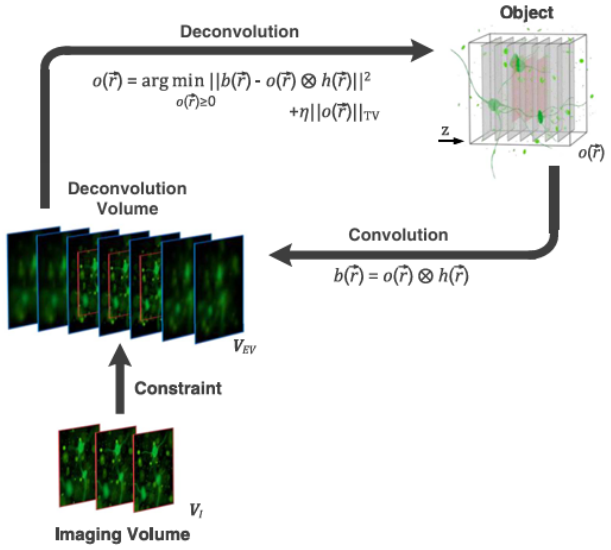


Figure 2. Schematic of EV-3D deconvolution algorithm proposed by xiao et al [12].

2.2. Extended-Volume 3D Deconvolution

Since z-splitter imaging inevitably contains out-of-focus background signals, background suppression and contrast enhancement are essential for improving image clarity. To address this, we apply the EV-3D deconvolution algorithm proposed by xiao et al [12], as illustrated in Fig. 2. This method accounts for far-out-of-focus fluorescence originating outside the imaging volume by incorporating it into the deconvolution process. The algorithm formulates a minimization problem involving the estimated image $o(\vec{r}) \geq 0$ and the observed raw image $b_0(\vec{r}) \geq 0$

$$\min_{o,b} f(o,b) = \min_{o(\vec{r}),b(\vec{r})} ||(o \otimes h)(\vec{r}) - b(\vec{r})||_2^2 + \eta||o(\vec{r})||_{\mathrm{TV}} \tag{1}$$

$$\text{s.t. } b(\vec{r}) = b_0(\vec{r}) \text{ for } \vec{r} \in \mathbf{V}_I \tag{2}$$

in cartesian coordinate $\vec{r} = (x, y, z)$ where $h(\vec{r})$ is the point spread function (PSF) of the system, $\mathbf{V}_I$ is the imaging volume before deconvolution, and $||o(\vec{r})||_{\mathrm{TV}} = \sum_{\vec{r} \in \mathbf{V}_{\mathrm{EV}}} \sqrt{(\partial o/\partial x)^2 + (\partial o/\partial y)^2}$ is the isotropic total variation norm in the extended volume $\mathbf{V}_{\mathrm{EV}}$ when the regularization parameter η is small but greater than zero.

Starting from $b^1(\vec{r} \notin \mathbf{V}_I) = 0$, the deconvolution proceeds via nested iterations to optimize the objective function, as described in Algorithms 1 and 2.

Algorithm 1: EV-3D Deconvolution

Initialize: $b^1 = b^0$
for $k = 1, 2, \cdots, K$ **do**
  $o^{k+1} = \mathrm{argmin}_{o \geq 0} f(o, b^k)$
  $b^{k+1} = \mathrm{argmin}_{b \geq 0} f(o^{k+1}, b),\ b(\vec{r} \in \mathbf{V}_I) = b_0$

Algorithm 2: Update of o at Iteration k

Initialize: $o_1 = b^k$
for $n = 1, 2, \cdots, N$ **do**
  $o_{n+1} = (\frac{b^k}{o_n \otimes h} \otimes h^*) \cdot \frac{o_n}{1 - \eta \mathrm{div}(\nabla o_n / |\nabla o_n|}$
  $o_{n+1} = \max(o_{n+1}, 0)$
return $o^{k+1} = o_{N+1}$

We gratefully acknowledge the authors of the z-splitter imaging research for providing the MATLAB code in their GitHub repository [11], which served as a reference for our implementation. However, we migrated the code to Python with *cupy* package (the GPU-accelerated counterpart of the well-known *numpy* library) to create an integrated Python project.

This EV-3D deconvolution algorithm demonstrates significantly faster performance when employing a 'warm start', by optimally initializing $b^1(\vec{r} \notin \mathbf{V}_I)$ (i.e. using projection onto the imaging volume). It is stated to result in the reduction of outer iteration to $K = 1$ with only a small degradation in image quality.

2.3. Volumetric Reconstruction

Although EV-3D deconvolution algorithm incorporates depth extension during computation, it is intended to ad-

dress far-out-of-focus background signals rather than to perform volumetric reconstruction. While full volumetric reconstruction can be roughly done with spline interpolation using Python *scipy* package, deep learning-based methods offer a more elastic approach for interpolating sparse z-stack images.

The 3D U-Net [1], first introduced in 2016, is still a good option for volumetric reconstruction. It is an extension of the original U-Net architecture [8] into three dimensions, featuring a characteristic U-shaped network with a contracting path for downsampling and an expanding path for upsampling, as shown in Figure 3. Originally developed as a fully convolutional network [4] primarily for segmentation tasks, U-Net can also be adapted for reconstruction by expanding the output shape of the network. This flexibility is enabled by the upsampling process, which transforms coarse feature maps into dense volumetric outputs of arbitrary shape.

Since there is no publicly available dataset of volumetric organelle images suitable to train networks, we generated synthetic three-dimensional artifacts resembling mitochondria and lysosomes by creating randomly bent cylinders and dented spheres. These objects are arbitrarily sliced to produce z-stacks as input for training, while their intact volumes serve as the expected network output.

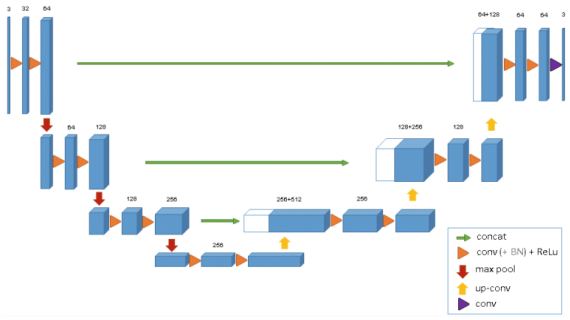


Figure 3. The 3D U-Net architecture proposed by ronneberger et al [8]. Blue boxes represent feature maps. The number of channels is denoted above each feature map.

2.4. Visualization

After the deconvolution and reconstruction, the resulting data is expected to be a three-dimensional array representing a single alpha-channel volumetric image. Three-dimensional visualization can be performed using Python *vedo* package, which is easy-use and supports basic interactions such as rotation and zooming. We also utilize Unity3D, a widely used game engine to enhance interactivity. The reconstructed data is converted from a 3D NumPy array to JSON (serialized text format), loaded into the Unity executable, and applied to construct voxel-based shapes, enabling highly interactive visualization complemented by a user-friendly interface.

3. Experiments

3.1. 3D U-Net Interpolation Model

To implement a robust 3D U-Net interpolation model to reconstruct full volumetric data from 6-plane z-stack images, we generated thousands of synthetic three-dimensional volumes composed of randomly dented spheres and capsules. They were designed to mimic the appearance of cellular organelles to be served as the training dataset. Each volume is represented as 3D *numpy* arrays, where each element denotes the alpha-channel value of a grayscale voxel space.

The overall structure is mathematically computed by by defining a high-alpha membrane surface of finite thickness surrounding a low-alpha interior. To resemble real biological samples more closely, random Perlin noise [3] is applied to both surface (as arbitrary membrane distortion) and interior (as random inner haze), at the expense of increased data computation time.

The training code is implemented using the *PyTorch* framework, and the 3D U-Net architecture is adopted without modification from the original design presented by ronneberger et al [8]. It includes a downsampling encoder with convolution blocks, a bottleneck layer, and an upsampling decoder with deconvolution blocks. The network accepts volumetric input data with the same dimensions as the output; therefore, the sparse 6-plane z-stack images should be roughly interpolated to match the required input shape.

Each training epoch is monitored in real time using the Python *tqdm* library. Based on empirical observation, the learning rate is adaptively reduced when the validation loss gets stuck over several iterations, using *ReduceLROnPlateau* scheduler.

Model with additional z-step input. Since the z-axis spacing (commonly referred to as the 'z-step') between image stacks is uniform but can take arbitrary values, we experimentally introduced a partial modification to the 3D U-Net architecture to incorporate the scalar z-step as an additional input parameter, alongside the interpolated volume. This is achieved by expanding the channel dimension of input volume and filling the second channel entirely with the z-step value, following the approach proposed by Morimoto et al. [6] To accommodate the change in input shape, the network architecture is slightly modified to accept double input channels.

The dataset consists of 1,200 dented capsule-like artifacts; 960 for training and 240 for testing, designed to mimic mitochondria. Each volume has a shape of $96 \times 96 \times 96$ voxels, with random membrane thickness, alpha values, and noise intensity. The initial learning rate is set to 1e-3,

and the model is trained for 100 epochs.

The training is conducted in a local Python 3.13.2 envionment on a desktop equipped with Intel i9-10900K CPU, 64GB RAM and NVIDIA GeForce RTX 3090 GPU. GPU operations have been supported with NVIDIA Game Ready driver 572.70 and corresponding CUDA compilation toolkit of version 12.8. Each training epoch took approximately 102 seconds, total 3 hours to complete the full training process and generate the final model checkpoint. The loss function combines the inverse Structural Similarity Index (1-SSIM) and L1 loss (mean absolute error, MAE) as defined in Equation 3. Using this composite loss function, the trained model achieved a final loss value of 0.0373 on the test dataset.

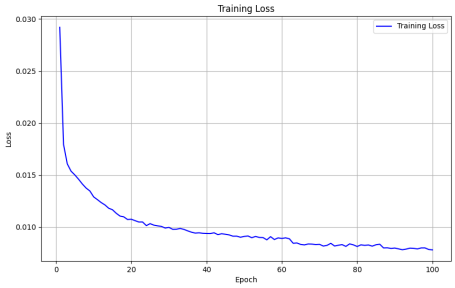
$$loss = \alpha \times (1 - \text{SSIM}) + (1 - \alpha) \times \text{L1} \quad (\alpha = 0.85) \quad (3)$$

As shown in Figure 4, the training process appears to have proceeded successfully, with a gradual decrease in loss over iterations and convergence toward a final value close to 0.003. When tested on redundant artifacts during validation inference, the reconstruction often yields results with reasonably low visual error. However, in several cases, the model fails to generate the expected organelle-like shapes, instead producing box-like artifacts or even entire vacancy.

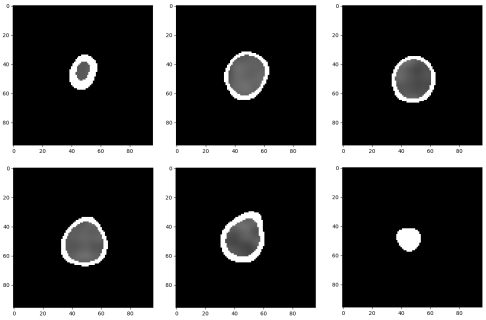
These failures, while unexpected, may be attributed to architectural limitations. Specifically, the 3D U-Net model used here is fundamentally designed to accept only single-channel input, which might restrict its ability to incorporate contextual or geometric cues provided by additional inputs, such as the z-step value. Although this hypothesis is difficult to verify analytically, it highlights a potential limitation of the standard 3D U-Net in handling sparse volumetric reconstruction tasks.

Final interpolation model. Following the unsuccessful implementation of the double-channel 3D U-Net architecture with z-step injection, we reverted to the standard single-channel 3D U-Net, assuming a constant z-step value across all input data. In this experiment, we shift the focus from capsule-like mitochondrial structures to spherical organelles, such as lysosomes. The new dataset comprises 1,800 synthetic dented sphere-like artifacts, with 1,440 samples used for training and 360 for testing. Each sample is represented as a $64 \times 64 \times 64$ volumetric array, created with randomized shape parameters.

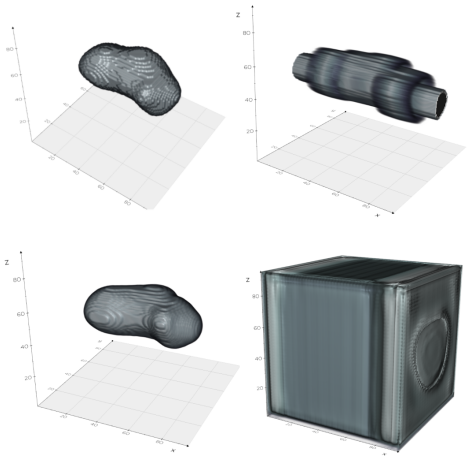
All hyperparameters, including the learning rate, optimizer settings, and training environment, are kept intact with the previous experiment. In this case, however, additional data augmentation are introduced, including random flips and additive Gaussian noise, to improve generalization and robustness to better deal with non-identical input data.



(a) The training loss over epochs.

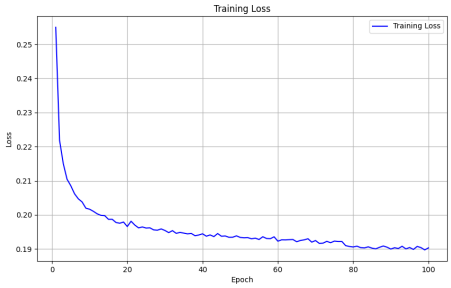


(b) An input z-stack data for validation, gathered from raw volumetric artifact shown in (c).

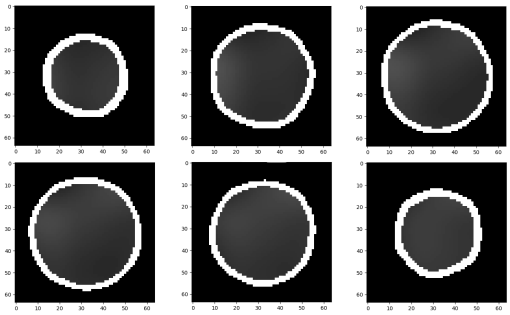


(c) An example of inference results is shown using the Python *vedo* package for grayscale volumetric visualization. (Top-left) The ground truth volume, a dented capsule-like synthetic artifact of size $96 \times 96 \times 96$, designed to mimic mitochondrial morphology. (Top-right) Roughly interpolated input volume generated from the original z-stack consisting of 6 slices. The shape is upsampled to match the input requirements of the 3D U-Net model. (Bottom-left) Output of the model after successful interpolation. The result closely approximates the ground truth, achieving MSE=0.005. (Bottom-right) An example of a failed reconstruction.

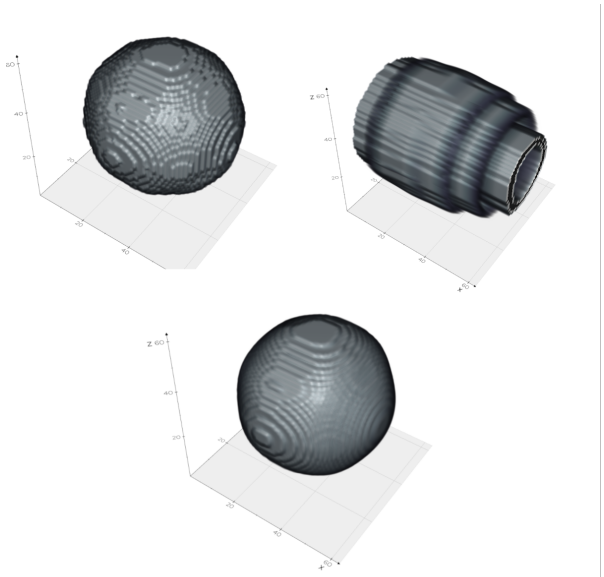
Figure 4. Training and validation of the 3D U-Net interpolation model with additional z-step input.



(a) The training loss over epochs.



(b) An input z-stack data for validation, gathered from raw volumetric artifact shown in (c).



(c) An example inference result from the validation set, visualized using the Python *vedo* package for grayscale volumetric rendering. (Top-left) The ground truth volume—a dented spherical artifact of size $64 \times 64 \times 64$, designed to mimic the morphology of lysosomes. (Top-right) The roughly interpolated volume derived from the z-stack consisting of 6 axial slices. The input is upsampled to match the input requirements of the 3D U-Net model. (Bottom) The reconstructed volume produced by the trained model, showing successful interpolation with a close visual approximation to the ground truth.

Figure 5. Training and validation of the final interpolation model.

The results are presented in Figure 5. Each training iteration took approximately 52 seconds, resulting in a total training time of around 1.5 hours to generate the model checkpoint. The training process has shown stable convergence of the loss function, although the final loss values are relatively higher compared to those observed in the previous model with double input channels.

This increase in loss is attributed to the computational characteristics of the loss function; in the double-channel setup, the additional z-step input introduced a constant-valued dimension that may have mathematically reduced the error. In contrast, the single-channel model lacks this auxiliary information, leading to slightly less constrained loss.

Despite this, the reconstruction from the single-channel model demonstrates low error, high stability, and consistent performance. Even with the introduction of expected extents of noise in the input data, the model rarely fails to generate coherent organelle-like shapes, confirming its robustness and effectiveness for this task.

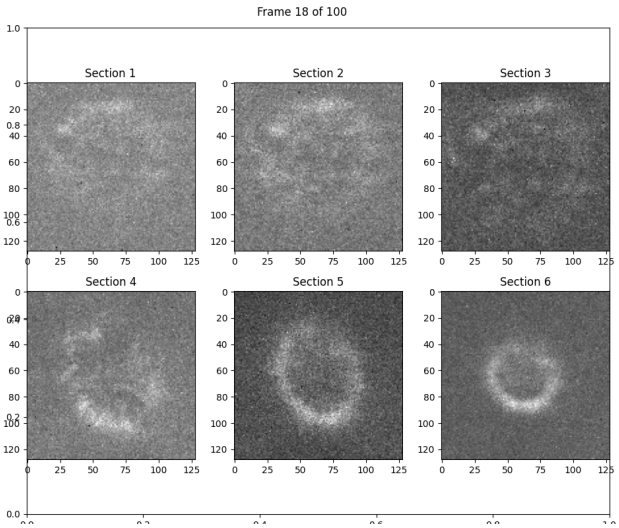
3.2. Nanobead Sample

To evaluate whether the interpolation model is suitable and efficient for application to real z-splitter microscopy images, we utilized a dataset consisting of 100 frames of six-plane image stacks of nanobead samples, recorded at 50 frames per second (Figure 1). Each image is loaded as a 2D grayscale array with pixel values normalized to the range $[0, 1]$ and a resolution of 128×128 pixels. For each frame, a circular object consistently present across all z-planes is identified, cropped for focus, and subjected to contrast enhancement using gamma correction with $\gamma = 5$, as shown in Figure 6.

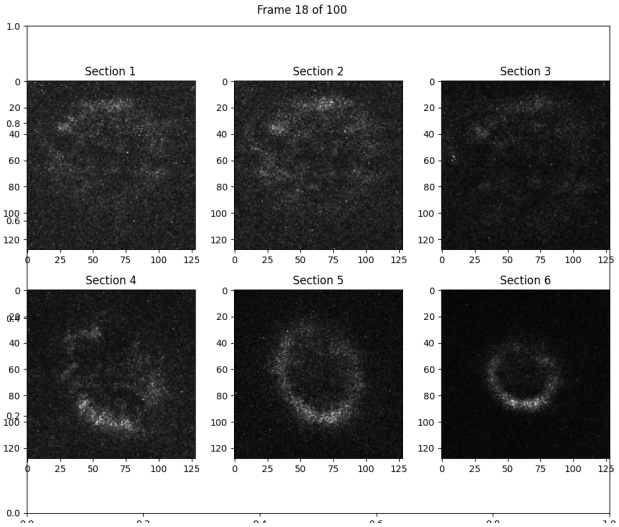
EV-3D deconvolution. The Python implementation of MatLab EV-3D deconvolution algorithm provided by xiao et al. [11] is applied to the contrast-enhanced z-stack images in this research. Based on the ratio between the actual sample size and the image dimensions, the lateral pixel size is set to $dx = 3.9\text{nm}$ and axial (z-step) interval $dz = 27.3\text{nm}$. The objective numerical aperture (NA) is slightly adjusted for the definition of deconvolved image, as $\text{NA} = 1.3$, a value suitable for super-resolution microscopy.

For the optimization parameters, the lateral and axial volume extension is set to $xy_{pad} = 10$ and $z_{pad} = 5$, respectively. The number of outer and inner iterations are fixed at $K = 20$ and $N = 80$, referring to the original MatLab implementation. Total variation (TV) regularization is not applied in this experiment.

Under the aforementioned local computing environment, the algorithm required approximately 2.2 seconds per frame (with each frame sized $128 \times 128 \times 6$), resulting in a total computation time of around 4 minutes for the 100-frame

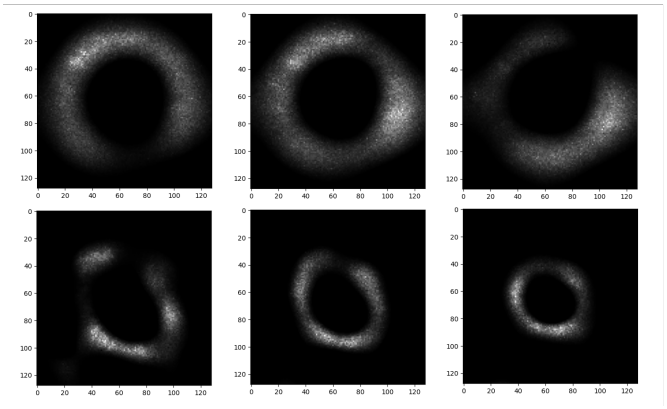


(a) Zoomed-in sections of the six z-stack images from a single frame. Since the circular object remains seemingly consistent across frames, its location was manually defined and fixed for all frames. Each section is handled as a 2D grayscale *numpy* array.

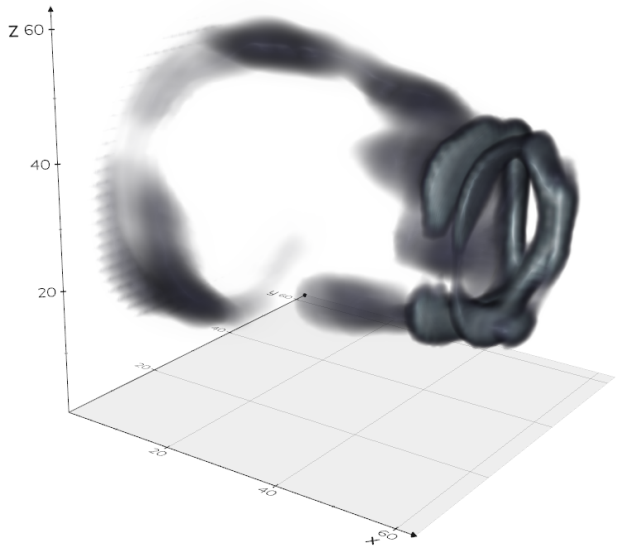


(b) Contrast-enhanced version of (a), produced by applying element-wise gamma correction with $\gamma = 5$.

Figure 6. The raw z-splitter images of circular imagery in a single frame of nanobead sample video.



(a) The result of EV-3D deconvolution with contrast-enhanced z-splitter images of nanobead sample (Figure 6-(b). It shows faint circular shapes by suppressing background noises.



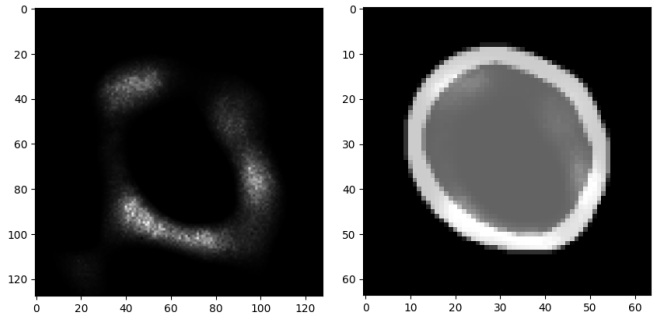
(b) The interpolation result with deconvolved z-splitter images (a). It fails to generate sphere-like shapes due to ambiguous border of the circular imagery in input images.

Figure 7. The interpolation fails with intact EV-3D deconvolved image stacks.

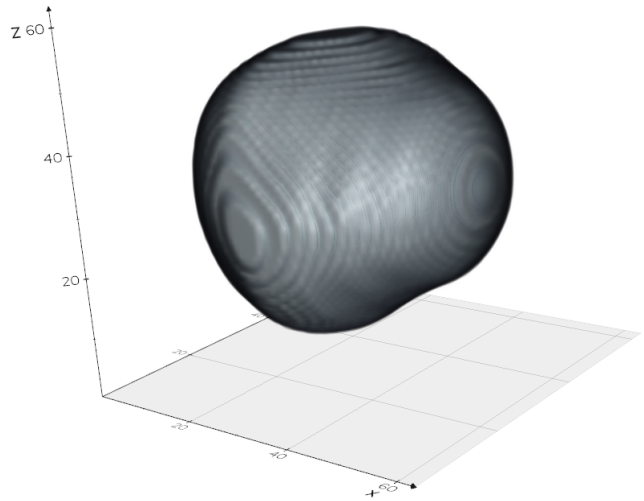
nanobead video dataset. The resulting deconvolved image stacks are displayed in Figure 7-(a). Since the interpolation model expects input stacks consisting of six 64×64 planes, the deconvolved stacks are simply downscaled by half along the lateral dimensions prior to model input.

Interpolation (Model inference). When the deconvolved image stacks are fed into the interpolation model, the inference fails to produce the expected sphere-like shape, as shown in Figure 7-(b). Although data augmentation with artificial noise is applied during training to enhance model robustness, the nanobead sample images differ significantly from the idealized artifacts used in the training and validation datasets.

Through trial and error, we identified that the major reason of failure is the ambiguous and poorly defined borders of the circular features in the deconvolved images. This observation suggests that a dedicated preprocessing step is



(a) (left) a single image from deconvolved z-stacks. (right) masked version of deconvolved image, using modified version of Hough transform technique supported by *opencv-python*.



(b) The interpolation result with masked z-splitter images. It gives successful interpolation result of imperfect spherical shapes.

Figure 8. The interpolation successfully gives spherical shapes with masked EV-3D deconvolved image stacks.

necessary to improve the interpolation performance, as outlined below:

The solution involves constructing a clear border using ring detection. Utilizing the *opencv-python* package, a modified version of the Hough transform with generous threshold parameters is applied to detect slightly imperfect circular shapes within the deconvolved stacks. Based on these detections, image masks shaped like the detected rings have been generated, with low alpha values assigned inside the ring boundaries.

For model inference, the input is replaced by a linear combination of these generated mask images and the original deconvolved images. This preprocessing enable the interpolation model to successfully produce spherical shapes, as demonstrated in Figure 8. On the local environment, the interpolation step took approximately 0.29 seconds per frame, resulting in a total of 30 seconds to generate the interpolated 3D video for all 100 frames of the fluorescent nanobeads. Adding the EV-3D deconvolution time, the entire computation required only 4.5 minutes.

3.3. Interactable Visualization with Unity3D

Since Python *vedo* visualization lacks advanced interactivity, we developed a simple visualization program using the Unity3D engine. The reconstructed nanobead sample data are exported in .json format and loaded into Unity, where the volumetric shape is composed of unit voxel cubes that can be freely rotated and zoomed.

An earlier attempt involved converting the data into an alpha-channel texture for use with a transparent raymarching HLSL shader; however, this approach suffered from distortion issues during rotation.

As shown in Figure 9, the Unity-based visualization includes additional user interface elements for playing frame-by-frame animations and viewing single cross sections (z-index slices) of the volumetric shapes.

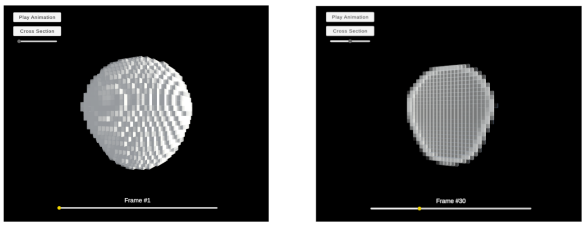


Figure 9. Three-dimensional visualization of reconstructed nanobead volumetric image, using Unity3D engine. It contains some user interfaces for interaction.

4. Conclusion

So far, we have introduced the volumetric reconstruction strategy for z-splitter images of cellular organelles, combining the EV-3D deconvolution algorithm for suppressing far-out-of-focus background and a 3D U-Net interpolation model to generate full volumetric shapes from sparse z-stacks. Taking the advantages of z-splitter imaging that is low-cost, easy to assemble, and adaptive to rapidly changing samples, our approach shows promise for further investigation.

Considering this research lacks real biological samples and professional environment, it is notable that the reconstruction produced seemingly successful result when applied to real nanobead images. Additionally, the strategy shows impressive time efficiency; the training procedure took only 1.5 hours on 1,440 artifacts, and the reconstruc-

tion requires just 2.7 seconds per frame. However as implied, it also has some limitations:

First, this research has a critical flaw in that it does not provide quantitative evaluation of recontruction, due to the absence of actual three-dimensional ground-truth data. For the nanobead experiments, only 2D image stacks have been available, with no corresponding volumetric reference for comparison. Therefore, the results cannot be regarded as scientifically validated, but rather as an indication of the potential of our approach, demonstrated by the generation of seemingly plausible spherical volumetric shapes.

Second, it would be quite challenging to create suitable datasets for model training. To develop a reliable 3D U-Net interpolation model for specific biological samples rather than our rough model trained with the rough artifacts, thousands of full volumetric images of those specific samples would be required. Given the difficulty of obtaining comprehensive 3D images of microorganisms, this approach may not be practical in many real-world scenarios.

Finally, an inherent limitation arises from the nature of z-splitter imaging itself. While it can effectively restore the surface morphology of samples, it cannot recover internal structures. This is because detailed internal information is irretrievably lost when decomposing the full three-dimensional volume into sparse z-stack images.

In summary, we have demonstrated the potential of applying a 3D U-Net interpolation model for the reconstruction of microscopic images gathered with a z-splitter prism. Although our research and strategy has several limitations, it shows promise with its visual quality and time efficiency, warranting further investigation.

References

[1] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II 19*, pages 424–432. Springer, 2016. 1, 3

[2] Florian O Fahrbach, Fabian F Voigt, Benjamin Schmid, Fritjof Helmchen, and Jan Huisken. Rapid 3d light-sheet microscopy with a tunable lens. *Optics express*, 21(18):21010–21026, 2013. 1

[3] Salakhiev Ildar. perlin-noise. https://github.com/salaxieb/perlin_noise, 2021. 3

[4] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 3

[5] Manuel Martı, Po-Yuan Hsieh, Ana Doblas, Emilio Sanchez-Ortiga, Genaro Saavedra, Yi-Pai Huang, et al. Fast axial-scanning widefield microscopy with constant magnification and resolution. *Journal of Display Technology*, 11(11):913–920, 2015. 1

[6] Masaki Morimoto, Kai Fukami, Kai Zhang, Aditya G Nair, and Koji Fukagata. Convolutional neural networks for fluid flow analysis: toward effective metamodeling and low dimensionalization. *Theoretical and Computational Fluid Dynamics*, 35(5):633–658, 2021. 3

[7] John M Murray. Methods for imaging thick specimens: confocal microscopy, deconvolution, and structured illumination. *Cold Spring Harbor Protocols*, 2011(12):pdb–top066936, 2011. 1

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 3

[9] Jean-Marc Tsang, Howard J Gritton, Shoshana L Das, Timothy D Weber, Christopher S Chen, Xue Han, and Jerome Mertz. Fast, multiplane line-scan confocal microscopy using axially distributed slits. *Biomedical Optics Express*, 12(3): 1339–1350, 2021. 1

[10] Yichen Wu, Yair Rivenson, Hongda Wang, Yilin Luo, Eyal Ben-David, Laurent A Bentolila, Christian Pritz, and Aydogan Ozcan. Three-dimensional virtual refocusing of fluorescence microscopy images using deep learning. *Nature methods*, 16(12):1323–1331, 2019. 1

[11] Sheng Xiao. Extended-volume-3d-deconvolution. https://github.com/shengxiao01/Extended-Volume-3D-deconvolution, 2020. 2, 5

[12] Sheng Xiao, Howard Gritton, Hua-an Tseng, Dana Zemel, Xue Han, and Jerome Mertz. High-contrast multifocus microscopy with a single camera and z-splitter prism. *Optica*, 7(11):1477–1486, 2020. 1, 2

[13] Dong-Jin Yoo. Three-dimensional surface reconstruction of human bone using a b-spline based interpolation approach. *Computer-Aided Design*, 43(8):934–947, 2011. 1

[14] Bingying Zhao, Minoru Koyama, and Jerome Mertz. High-resolution multi-z confocal microscopy with a diffractive optical element. *Biomedical Optics Express*, 14(6):3057–3071, 2023. 1