Optics Letters

Fast algorithm for 3D volume reconstruction from light field microscopy datasets

Jonathan M. Taylor*

School of Physics and Astronomy, University of Glasgow, Glasgow G12 8QQ, UK
*jonathan.taylor@glasgow.ac.uk

Received 20 March 2023; revised 5 July 2023; accepted 5 July 2023; posted 6 July 2023; published 2 August 2023

Light field microscopy can capture 3D volume datasets in a snapshot, making it a valuable tool for high-speed 3D imaging of dynamic biological events. However, subsequent computational reconstruction of the raw data into a human-interpretable 3D+time image is very time-consuming, limiting the technique's utility as a routine imaging tool. Here we derive improved equations for 3D volume reconstruction from light field microscopy datasets, leading to dramatic speedups. We characterize our open-source Python implementation of these algorithms and demonstrate real-world reconstruction speedups of more than an order of magnitude compared with established approaches. The scale of this performance improvement opens up new possibilities for studying large timelapse datasets in light field microscopy.

https://doi.org/10.1364/OL.490061

Light field microscopy provides snapshot 3D imaging of dynamic scenes *via* a lenslet array placed in the image plane of the microscope, which casts a multi-apertured intensity pattern onto a camera sensor. The mix of spatial and angular information about the target sample emission in this single raw 2D snapshot image allows 3D image reconstruction across an extended depth-of-field [Fig. 1(a)], but only after intensive computational processing [1]. Solving this inverse problem is extremely computationally demanding, traditionally requiring the computation of thousands of Fourier transforms (FTs) per iteration. This requirement for high levels of computational resource is particularly problematic given the attractiveness of light field imaging for 3D time series [2–5], where a large number of different timepoints must all be reconstructed.

Here we will demonstrate how to mathematically simplify the light field reconstruction process, speeding up real-world computation times by more than an order of magnitude, while delivering identical output volume results.

The image reconstruction process is an inverse problem that can be cast as a deconvolution. The spatially variant point spread function (PSF) of the light field microscope is typically computed theoretically from wave-optics calculations [1,2], and Richardson–Lucy deconvolution is then used to estimate the 3D volume that gives rise to the measured 2D intensity pattern observed on the camera sensor. The standard Richardson–Lucy algorithm is described by the following iterative formula:

$$O_{\text{est}}^{(i+1)} = O_{\text{est}}^{(i)} \times H^T \left(\frac{I}{H(O_{\text{est}}^{(i)})} \right), \tag{1}$$

although the widely used light field implementation in [2] uses an alternative variation (see [6] for further discussion), where the error term is computed in object space:

$$O_{\text{est}}^{(i+1)} = O_{\text{est}}^{(i)} \times \frac{H^T I}{H^T H(O_{\text{est}}^{(i)})}. \tag{2}$$

In either form, $\times$ denotes elementwise multiplication and the fraction implies elementwise division; $O_{\text{est}}^{(i)}$ is the estimation of the 3D object to be reconstructed, at iteration i; I is the 2D light field image recorded on the camera; H is the "forward projection" operator mapping from the object O to the resultant camera image I; and H^T is the matrix transpose of the operator H. The optical interpretation of H^T leads to it being termed the backward-projection operator. A typical starting condition would be $O_{est}^{(0)} = H^T I$, and O_{est} converges to an estimate of the true object O over $N_{\text{iter}} \sim 10$ iterations.

The basic building blocks of the deconvolution problem are therefore the forward- and backward-projection operators, H and H^T, which model the image formation process. In light field microscopy, these projection operators are expressed as the sum of many separate convolution operations. Given a three-dimensional object O consisting of voxels indexed O_{xyz}, each pixel value I_{mn} of a forward-projected image I can be computed as

$$I_{mn} = \sum_{xyz} O_{xyz} \times H_{mnxyz}, \tag{3}$$

where H_{mnxyz} are the matrix elements of the PSF applicable to voxel x, y, z. To render the reconstruction problem tractable, raw images are resampled such that the footprint of each lenslet in the lenslet array spans an exact odd integer number of camera pixels, A. Thus, any subpixel a, b at the same relative position within any lenslet footprint will have the same the point spread function [Fig. 1(b)]. This simplifies the problem, enabling Eq. (3) to be rewritten as

$$I = \sum_{abz} M_{ab}\{O_z\} \otimes H_{abz}, \tag{4}$$

where M_{ab} is a masking operator which zeroes all pixels except those in the image pixel group satisfying $x \bmod A = a$ and

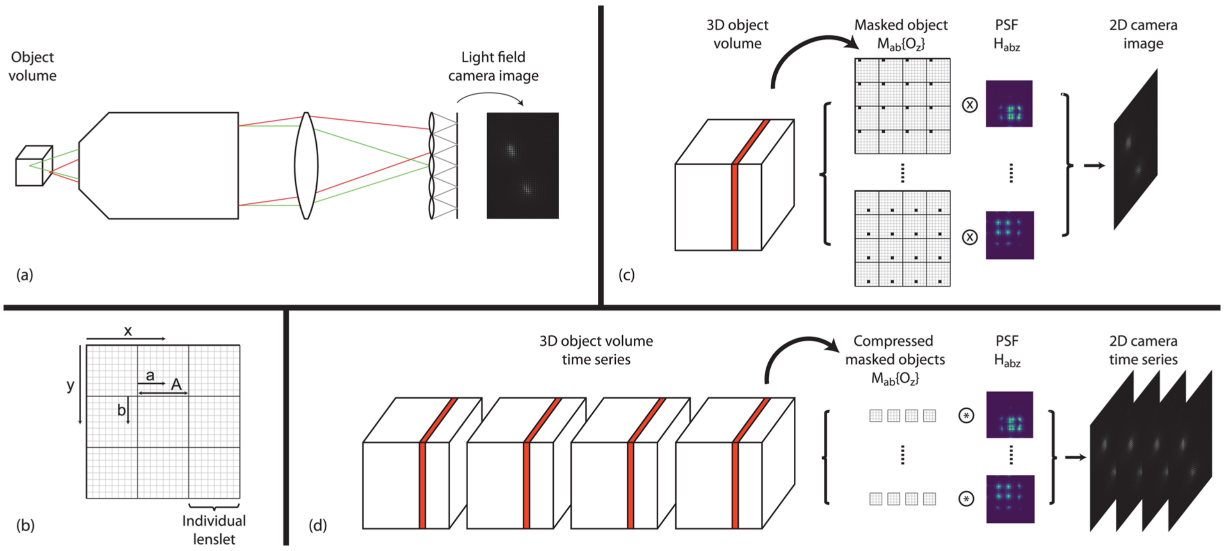

Fig. 1. (a) Optical schematic of light field microscopy. (b) Pixel indexing relative to lenslet footprints. (c) Conventional projection operation for deconvolution ($\otimes$ represents convolution). (d) New fast projection strategy ($\circledast$ represents a new custom operation—see main text).

$y \bmod A = b$, and $\otimes$ is the convolution operator. Here, H_{abz} represents the complete PSF for a voxel at coordinate a, b, z. Thus, the forward-projected image from an object consisting of Z individual z-planes can be computed using a total of A^2Z convolution operations.

According to the convolution theorem, each convolution consists of two FTs and one inverse, each computed using the fast Fourier transform (FFT) algorithm:

$$M_{ab}\{O_z\} \otimes H_{abz} = \mathcal{F}^{-1}\left(\mathcal{F}(M_{ab}\{O_z\}) \times \mathcal{F}(H_{abz})\right). \quad (5)$$

Any strategy aiming for more than merely incremental performance speedup of the overall calculation must speed up all three of the FFT operations in this equation. Speeding up any one of these on its own would be insufficient: even if the run time for one of them on its own could be reduced to zero, the overall run time would still only be improved by a factor of $\sim 30\%$. In what follows, we will demonstrate how to improve the run time of each of these three operations in turn to achieve an order of magnitude speedup in computation time. Our strategy is illustrated schematically in Figs. 1(c) and 1(d).

The discrete FT of the masked object $M_{ab}\{O_z\}$ involves a high degree of redundancy, since most elements of this masked object array are zero. We observe that this problem can be simplified by generalizing the Danielson–Lanczos lemma [7,8] to an A-way result. In one dimension, the kth element F_k of the discrete FT of a function f can be expressed in terms of A smaller FTs:

$$F_k = \tilde{F}_k^{(0)} + (W)^k \tilde{F}_k^{(1)} + (W)^{2k} \tilde{F}_k^{(2)} + \cdots + (W)^{(A-1)k} \tilde{F}_k^{(A-1)}, \quad (6)$$

where $W = \exp(-2\pi i/X)$ and $\tilde{F}_k^{(a)}$ is the (reduced-size) discrete FT of $\tilde{f}$, where $\tilde{f}$ is a vector consisting of only the nonzero elements of $M_a\{f\}$. Note that, in the notation of Eq. (6), k indexes the reduced-size discrete FT $F_k^{(a)}$ beyond its normal domain of $k \in [0, X/A)$, exploiting the periodic boundary conditions implicit in the FT.

In our case, the M operator ensures that only one of the A distinct terms on the right-hand size of Eq. (6) is nonzero. Therefore, eliminating all the zero terms, generalizing to two dimensions, and summing over all image pixel groups a, b:

$$F_{mnz} = \sum_{ab} (W)^{abmn} \tilde{F}_{mnz}^{(a,b)}. \quad (7)$$

Consequently, instead of requiring A^2 FFTs that each operate over the full image size XY, we have reduced these to operating on arrays of size XY/A^2 (compressed arrays representing image pixel groups each containing only those pixels retained by the $M_{ab}\{O_z\}$ operator). We have therefore reduced the computational requirements of these FFTs by a factor of A^2. After computation of the FFTs, the weighting multiplications in Eq. (7) must still be applied, albeit in an operation of reduced computational complexity $O(XY)$, but overall, the computational demands of computing F_{nm} are dramatically reduced.

We now move on to consider the FT of the point spread function H_{abz}. Here, H and its FTs are invariant properties of the imaging system. If sufficient memory storage is available, the FTs can be precomputed once and the computation of $\mathcal{F}(H_{abz})$ eliminated completely from Eq. (4). However, given typical values of $A \geq 15$, $Z \sim 50$, and megapixel images, tens or hundreds of gigabytes of memory would be required to cache all the precomputed values. That may be feasible on some high-end CPU-based platforms, but exceeds the capacity of most GPU platforms. Nevertheless, even when precalculation is not possible, our algorithm amortizes the computation of each FT across batches of multiple timepoints in a time series dataset, reconstructing each batch concurrently. We also exploit symmetry relationships in the PSFs (as also used in [9]), to permit rapid computation of e.g., $\mathcal{F}(H_{(A-a)bz})$ once $\mathcal{F}(H_{abz})$ has been computed on-the-fly.

Finally, by explicitly substituting Eq. (5) into Eq. (4) and exploiting the distributive property of the FT to promote the inverse FT outside the summation over a and b, we can dramatically reduce the number of inverse FTs required:

$$I_{mn} = \sum_z \mathcal{F}^{-1} \sum_{ab} \mathcal{F}(M_{ab}\{O_z\}) \times \mathcal{F}(H_{abz}). \quad (8)$$

We note that in a practical implementation using a non-circulant FT, it is not feasible to further promote the inverse FFT outside the summation over z, because the size of the intermediate arrays will vary according to the lateral extent of H_{abz}, and hence vary with z.

The backward-projection operator required for Richardson–Lucy deconvolution is traditionally denoted H^T in recognition that it is the transpose of the matrix operator H. However, in practice, H and H^T are both implemented as convolutions [as per Eqs. (3) and (4)]. Hence, we follow an almost identical approach for the backward projection, starting from the following equation in place of Eq. (4):

$$O_{mnz} = \sum_{ab} M_{ab}\{I\} \otimes H^T_{abz}. \tag{9}$$

We note that during the one-time computation of H and H^T during initial optical modeling of the microscope PSF, computing H^T does not require additional convolution operations as used in [2], and can instead be populated near-instantaneously by pixelwise reshuffling of elements of H.

To achieve our anticipated order-of-magnitude speedup by using Eq. (7) to compute $\mathcal{F}(M_{ab}\{O_z\})$ in Eq. (5), it was necessary to write carefully optimized custom computer code, since the specific multiplication and tiling process underpinning a practical implementation of Eq. (7) is a performance bottleneck, and it is a highly bespoke operation that is not available in standard numerical libraries. We have made available a high-performance open-source reference implementation of our complete light field reconstruction algorithm [10]. It is written primarily in the Python programming language, with performance-critical code written in C++, Cython [11], and CUDA, drawing on the FFTW library [12] to compute FTs in our C++ code. To ensure maximum performance, our code incorporates elements of dynamic load-balancing between CPU cores, and machine-adaptive runtime optimizations.

Figure 2 presents performance benchmarks for our light field reconstruction code. Optimal performance is achieved for large batch sizes, in which scenario our new approach delivers a performance gain of 14× (CPU) and 34× (GPU) compared with the widely used MATLAB implementation [2] based on Eqs. (2), (4), and (9). We also compare with [1,13] (GPU-only) since this is faster than [2]: our code still achieves a 10× performance gain relative to [13]. All performance measurements were performed on a representative light field imaging scenario: 1463 × 1273 pixel frames from a dataset in [4] (full test parameters specified in [10]). CPU: 8 core Intel Xeon, 3 GHz, 32-GB RAM. GPU: PNY Quadro RTX A4000, 1.56 GHz, 16-GB RAM, 224 GB/s.

As explained above, our implementation performs optimally when batch-reconstructing multiple timepoints in a time series dataset simultaneously—although even for a single timepoint, our implementation already outperforms existing implementations. Figure 2 reveals the clear linear-plus-baseline relationship between overall run time and batch size for our code. Regardless of batch size, a fixed amount of computational work must be performed to compute $\mathcal{F}(H)$. This baseline work can only be eliminated completely if sufficient RAM is available to cache pre-calculated values for all $\mathcal{F}(H)$, which can extend to hundreds of gigabytes. The total additional work [computing the other elements of Eq. (4) via Eqs. (7) and (8)] scales linearly with the batch size. This vindicates our strategy of amortizing the constant work across simultaneous deconvolutions of multiple timepoints.

In our GPU implementation, we measure that the linear-scaling work is more substantial compared with the baseline

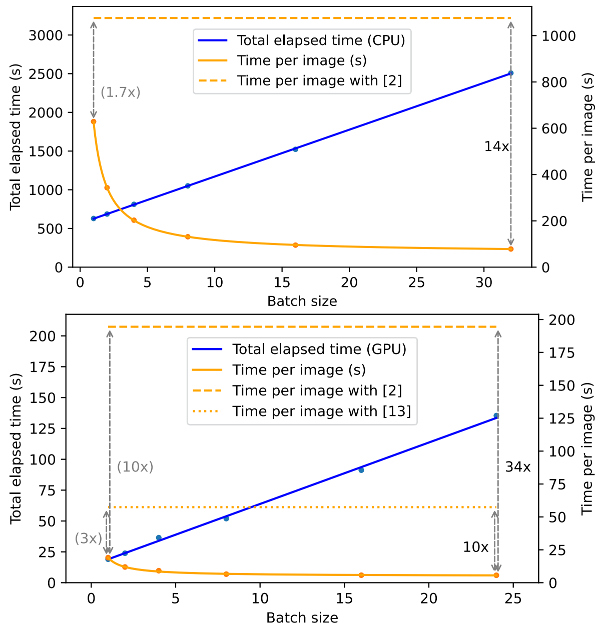

Fig. 2. Performance scaling with batch size. Blue plots (total elapsed time) show constant baseline work [computing $\mathcal{F}(H)$] plus linear scaling of additional work per batch item. Yellow plots (time per image) compare performance against previously published codes. As explained in the main text, our algorithms are designed to perform optimally with large batch sizes. Datapoints are measured run times, and trendlines are a fit to a linear scaling model.

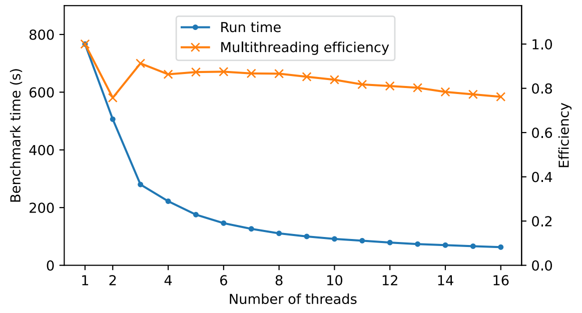

Fig. 3. Performance scaling with number of parallel CPU threads showing run-time improvement with increasing number of parallel threads. Multithreading efficiency is defined as 1.0 for a case where the measured time for an n-thread scenario is n times as fast as the 1-thread time.

work of computing $\mathcal{F}(H)$, probably due to the challenges of developing optimized CUDA kernels to implement the highly specific custom operations embodied in Eq. (7). This means that there are diminishing additional benefits to increasing the batch size beyond 8 on a GPU, which happily in turn means the RAM requirements are lower on a GPU, where RAM is typically scarcer.

Figure 3 confirms that our implementation scales well when the work is parallelized across multiple CPU cores: the benchmark run time with 16 parallel threads was over 12× faster than the single-threaded run time. The small decrease in efficiency (i.e., speedup being slightly less than n times when using n threads) can be explained by the increased pressure that multithreaded code imposes on the system's memory bandwidth. The slight anomaly in efficiency for 2 threads is likely related to the dual-CPU architecture of the testbed system.

We emphasize that our approach is a faster method to generate reconstructed volumes that are mathematically identical to the output of [2], which is widely accepted as a reference MATLAB implementation for light field volume reconstruction. Our benchmarking code includes verification that the structural similarity index (SSIM) between our reconstructions and those of [2] is 1.000000. A number of other light field reconstruction codes exist, many of which build on [2] by introducing additional features such as filtering to alleviate native focal plane artifacts. Our methodology should be directly transferable to most iterative codes for reconstructing light field datasets. It is beyond the scope of this paper to rewrite all these codebases, but in [14] we demonstrate how to interface our implementation with the existing MATLAB codebase of [2]. For other codes, the main computational work remains the forward and backprojection operations, so the scope for speed-up can be estimated by comparing their run-times to our benchmark results in Fig. 2. For example, Ref. [9] takes 39× longer (CPU) than our implementation. The increased speed of our approach also makes it realistic to consider the benefits of direct deconvolution of multi-view light field datasets, in contrast to the two-stage “reconstruct then fuse" approach taken in [4].

It is not practical to compare our code performance directly with phase-space deconvolution [15] due to the significant differences in reconstruction strategy (the respective merits of which are outside the scope of our work presented here). Their code does not scale well to the size of our test case (it would require 130 GB of RAM), but extrapolation from smaller test cases suggests our new code outperforms it significantly in terms of run time. It may well be possible to transfer elements of our approach to benefit the run time of [15], but further study would be required.

While many researchers continue to prefer the mathematically precise reconstruction afforded by the classical approach of Eq. (2), others are researching the use of machine-learning to estimate the object from the raw camera image [5,16–19]. The aim of such research is to compute an object estimate that is as faithful as possible to the true object, while using less computational time than required to explicitly compute Eq. (2). Some approaches commence with a limited number of Richardson–Lucy iterations [17] or use direct optical modeling during the training phase [5]; computational performance in both these scenarios will benefit directly from our new results. Pure machine-learning based approaches also stand to benefit from our results: machine-learning architectures in imaging are commonly based around convolutional neural networks, and it is increasingly recognized that networks perform better and can be trained faster when the network structure encapsulates physical insights into the image formation process [18,19]. Thus, the mathematical insights behind our results also hold promise for improving performance and effectiveness of future machine-learning architectures for light field microscopy reconstruction.

In summary, we have presented mathematical results enabling a dramatic reduction in computational work for 3D image reconstruction in light field microscopy, without any detriment at all to the quality of the reconstruction. The results and approach we have presented here are potentially applicable to any multi-aperture computational imaging system with a space-variant PSF endowed with translational symmetry properties. We have made available an open-source implementation of our algorithms [10], with performance measured to be more than an order of magnitude faster than previously available codes. We anticipate that our algorithms and open-source implementation will lead to an expansion in the uptake of light field microscopy as a tool for 3D+time imaging of rapid biological processes, now that the excessive computational demands of the reconstruction process have been brought under control.

Funding. Royal Society of Edinburgh (Sabbatical Grant).

Disclosures. The author declares no conflicts of interest.

Data availability. The computer code and test datasets underlying the results presented in this paper are available in Refs. [10,14].

REFERENCES AND NOTES

1. M. Broxton, L. Grosenick, S. Yang, N. Cohen, A. Andalman, K. Deisseroth, and M. Levoy, Opt. Express **21**, 25418 (2013).
2. R. Prevedel, Y. G. Yoon, M. Hoffmann, N. Pak, G. Wetzstein, S. Kato, T. Schrödel, R. Raskar, M. Zimmer, E. S. Boyden, and A. Vaziri, Nat. Methods **11**, 727 (2014).
3. O. Skocek, T. Nöbauer, L. Weilguny, F. Martínez Traub, C. N. Xia, M. I. Molodtsov, A. Grama, M. Yamagata, D. Aharoni, D. D. Cox, P. Golshani, and A. Vaziri, Nat. Methods **15**, 429 (2018).
4. N. Wagner, N. Norlin, J. Gierten, G. D. Medeiros, B. Balázs, J. Wittbrodt, L. Hufnagel, and R. Prevedel, Nat. Methods **16**, 497 (2019).
5. Z. Wang, L. Zhu, H. Zhang, G. Li, C. Yi, Y. Li, Y. Yang, Y. Ding, M. Zhen, S. Gao, T. K. Hsiai, and P. Fei, Nat. Methods **18**, 551 (2021).
6. In light field microscopy, native focal plane artifacts arise from missing structural information in the captured raw image. However, these are further exacerbated by the presence of background or scattered light outside the footprint of circular lenslets. Empirically, the performance of the alternative form [Eq. (2)] degrades less severely in this scenario, although we are not aware of any explicit mention or derivation of this property in the literature.
7. G. C. Danielson and C. Lanczos, J. Franklin Inst. **233**, 365 (1942).
8. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C* (Cambridge University Press, 1992).
9. A. Stefanoiu, J. Page, P. Symvoulidis, G. G. Westmeyer, and T. Lasser, Opt. Express **27**, 31644 (2019).
10. J. M. Taylor, “Python code for fast light field reconstruction,” Zenodo (2023), https://doi.org/10.5281/zenodo.7981141.
11. S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, Comput. Sci. Eng. **13**, 31 (2011).
12. M. Frigo, SIGPLAN Not. **34**, 169 (1999).
13. G. Schlafly, A. Verma, G. Harris, J. P. Vizcaino, P. L. Riviere, and R. Oldenbourg, “Light field imaging plugin for napari,” GitHub (2023), https://github.com/polarizedlightfieldmicroscopy/napari-lf.
14. J. M. Taylor, “Matlab/Python integration for fast light field reconstruction,” Zenodo (2023), https://doi.org/10.5281/zenodo.7736536.
15. Z. Lu, J. Wu, H. Qiao, Y. Zhou, T. Yan, Z. Zhou, X. Zhang, J. Fan, and Q. Dai, Opt. Express **27**, 18131 (2019).
16. N. Wagner, F. Beuttenmueller, N. Norlin, J. Gierten, J. C. Boffi, J. Wittbrodt, M. Weigert, L. Hufnagel, R. Prevedel, and A. Kreshuk, Nat. Methods **18**, 557 (2021).
17. Y. Zhang, B. Xiong, Y. Zhang, Z. Lu, J. Wu, and Q. Dai, Light: Sci. Appl. **10**, 152 (2021).
18. D. Deb, Z. Jiao, A. B. Chen, M. Broxton, M. B. Ahrens, K. Podgorski, and S. C. Turaga, “FourierNets enable the design of highly non-local optical 376 encoders for computational imaging,” arXiv, arXiv:2104.10611 (2022) .
19. Y. Li, Y. Su, M. Guo, X. Han, J. Liu, H. D. Vishwasrao, X. Li, R. Christensen, T. Sengupta, M. W. Moyle, I. Rey-Suarez, J. Chen, A. Upadhyaya, T. B. Usdin, D. A. Colón-Ramos, H. Liu, Y. Wu, and H. Shroff, Nat. Methods **19**, 1427 (2022).