

Supplementary Information for

Leonardo: a toolset to remove sample-induced aberrations in light sheet microscopy images

Yu Liu^{1,*}, Gesine F. Müller^{2,*}, Lennart Kowitz^{3,*}, Tomáš Chobola^{1,4}, Kurt Weiss⁵, Paul Maier², Jie Luo^{6,7,8,9}, Malte Roeßing³, Martin Stenzel³, Anika Grüneboom³, Johannes Paetzold⁶, Ali Ertürk^{6,7,8,10}, Nassir Navab^{11,12}, Carsten Marr¹³, Jianxun Chen^{3,†}, Jan Huiskens^{2,5,14,†}, Tingying Peng^{1,4,13,†}

¹ School of Computation, Information and Technology, Technical University of Munich, Munich, Germany

² Multiscale Biology, Department of Biology and Psychology, Georg-August-University Göttingen, Göttingen, Germany

³ Leibniz-Institut für Analytische Wissenschaften – ISAS – e.V., Dortmund, Germany

⁴ Helmholtz AI, Helmholtz Munich, Neuherberg, Germany

⁵ Morgridge Institute for Research, Madison, WI, USA

⁶ Institute for Intelligent Biotechnologies (iBIO), Helmholtz Center Munich, Neuherberg, Germany

⁷ Institute for Stroke and Dementia Research, Klinikum der Universität München, Ludwig-Maximilians University, Munich, Germany

⁸ Munich Cluster for Systems Neurology (SyNergy), Munich, Germany

⁹ Department of Industrial and Molecular Pharmaceutics, Purdue University, 575 Stadium Mall Drive, West Lafayette, IN 47907, USA

¹⁰ School of Medicine, Koç University, İstanbul, Turkey

¹¹ Computer Aided Medical Procedures, Technical University of Munich, Munich, Germany

¹² Computer Aided Medical Procedures, Johns Hopkins University, Baltimore, USA

¹³ Institute of AI for Health, Helmholtz Munich, Neuherberg, Germany

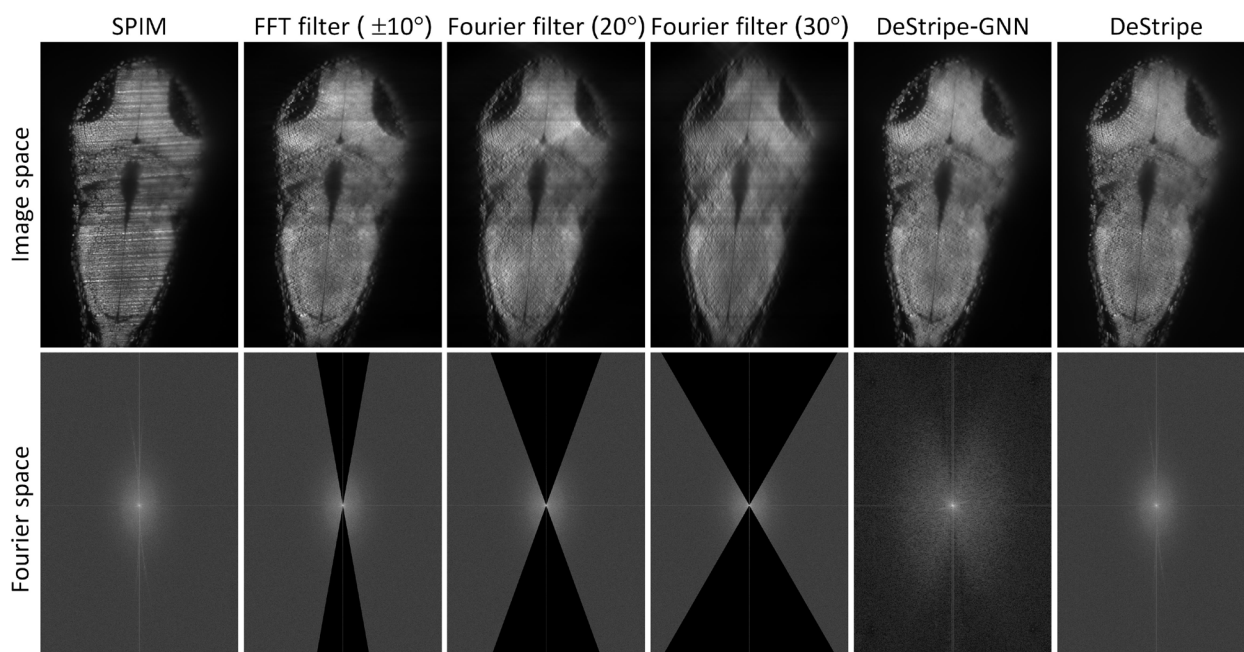
¹⁴ Cluster of Excellence “Multiscale Bioimaging: from Molecular Machines to Networks of Excitable Cells” (MBExC), University of Göttingen, Germany

* These authors contributed equally, † Corresponding authors

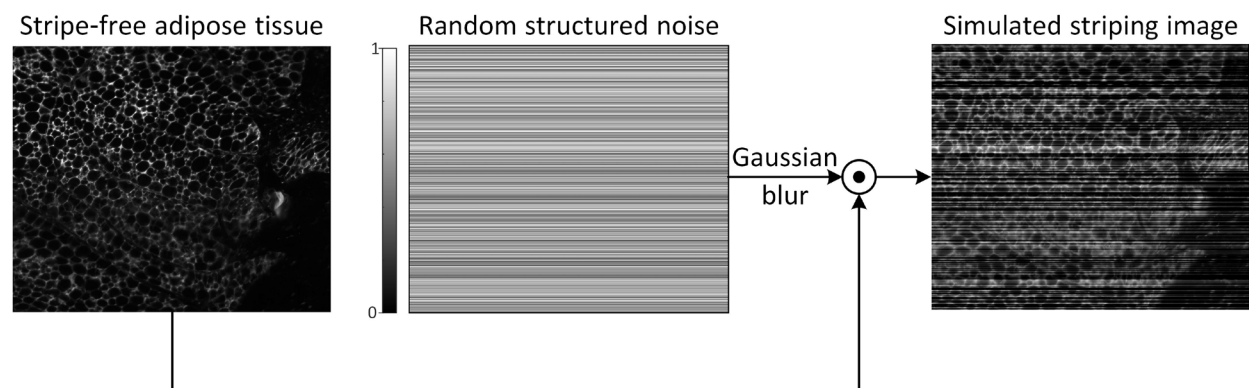
Content

Supplementary Figures	3
Supplementary Fig. 1	3
Supplementary Fig. 2	4
Supplementary Fig. 3	5
Supplementary Fig. 4	6
Supplementary Fig. 5	7
Supplementary Fig. 6	8
Supplementary Fig. 7	9
Supplementary Fig. 8	10
Supplementary Fig. 9	11
Supplementary Fig. 10	12
Supplementary Fig. 11	13
Supplementary Fig. 12	14
Supplementary Notes	15
Supplementary Note 1 Leonardo-DeStripe	15
Supplementary Note 1.1 Stripe remover regularized by anisotropic total variation	15
Supplementary Note 1.2 Improvement of Leonardo-DeStripe from Bayesian perspective	17
Supplementary Note 1.3 Guided filtering with various parameters	17
Supplementary Note 1.4 Leonardo-DeStripe is rotatable for stripes with arbitrary directions	18
Supplementary Note 2 Simulation of striping objects	21
Supplementary Note 3 Simulation of SPIM datasets with sequential dual-sided illumination	22
Supplementary Note 4 Information content assessment	23
Supplementary Note 5 SPIM registration	23
Supplementary Note 6 Optimized Leonardo-Fuse for large tissue	25
Supplementary Note 7 Leonardo-DeStripe-Fuse	25

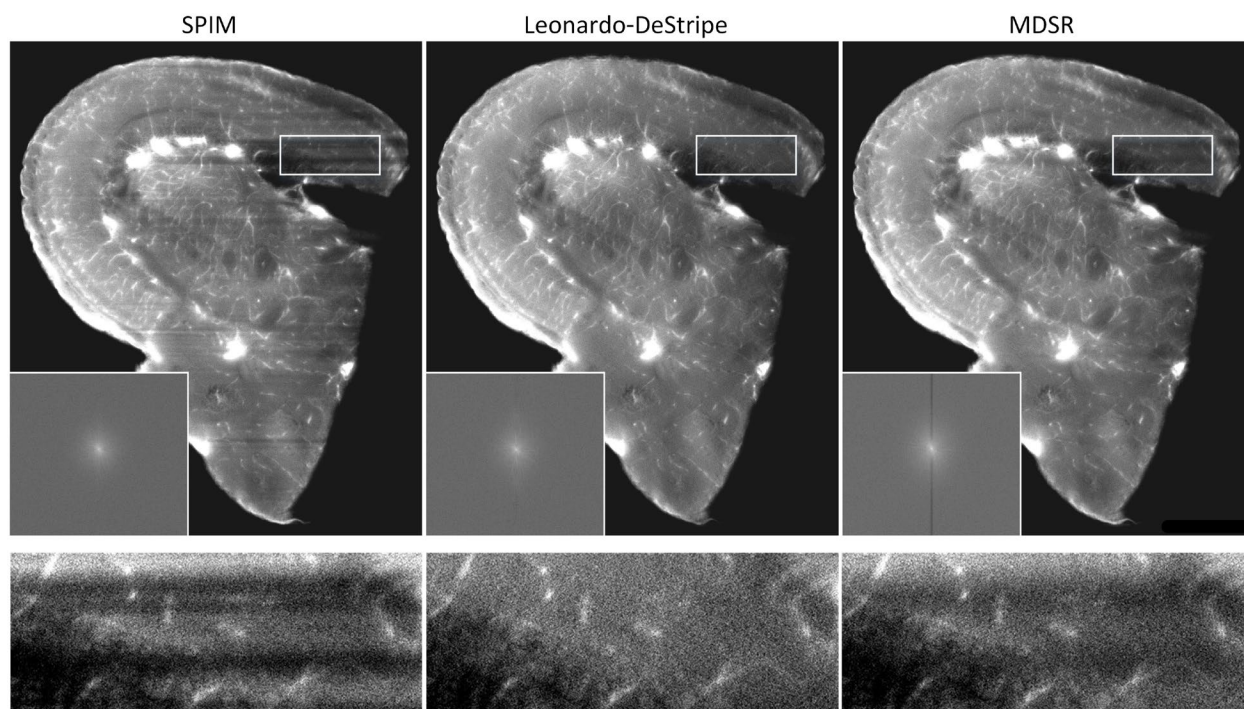
Supplementary Figures



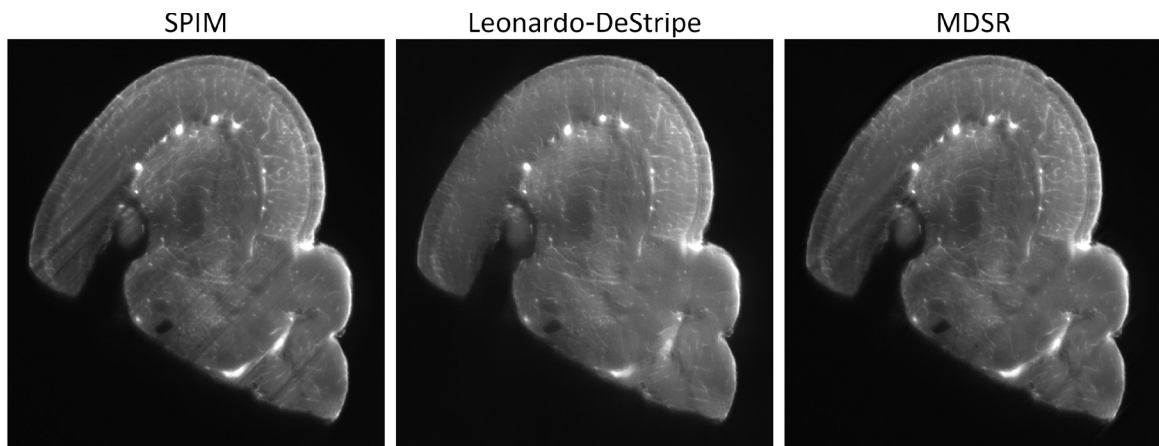
Supplementary Fig. 1. A comparison between Leonardo-DeStripe and conventional 2D band-pass FFT filter with different angular coverages of the wedge-shaped mask. From left to right, the angular coverage of the masking region in the FFT filter increases from $\pm 9^\circ$ to $\pm 29^\circ$. Note that the GNN in Leonardo-DeStripe is operated after downsampling (ratio of 3).



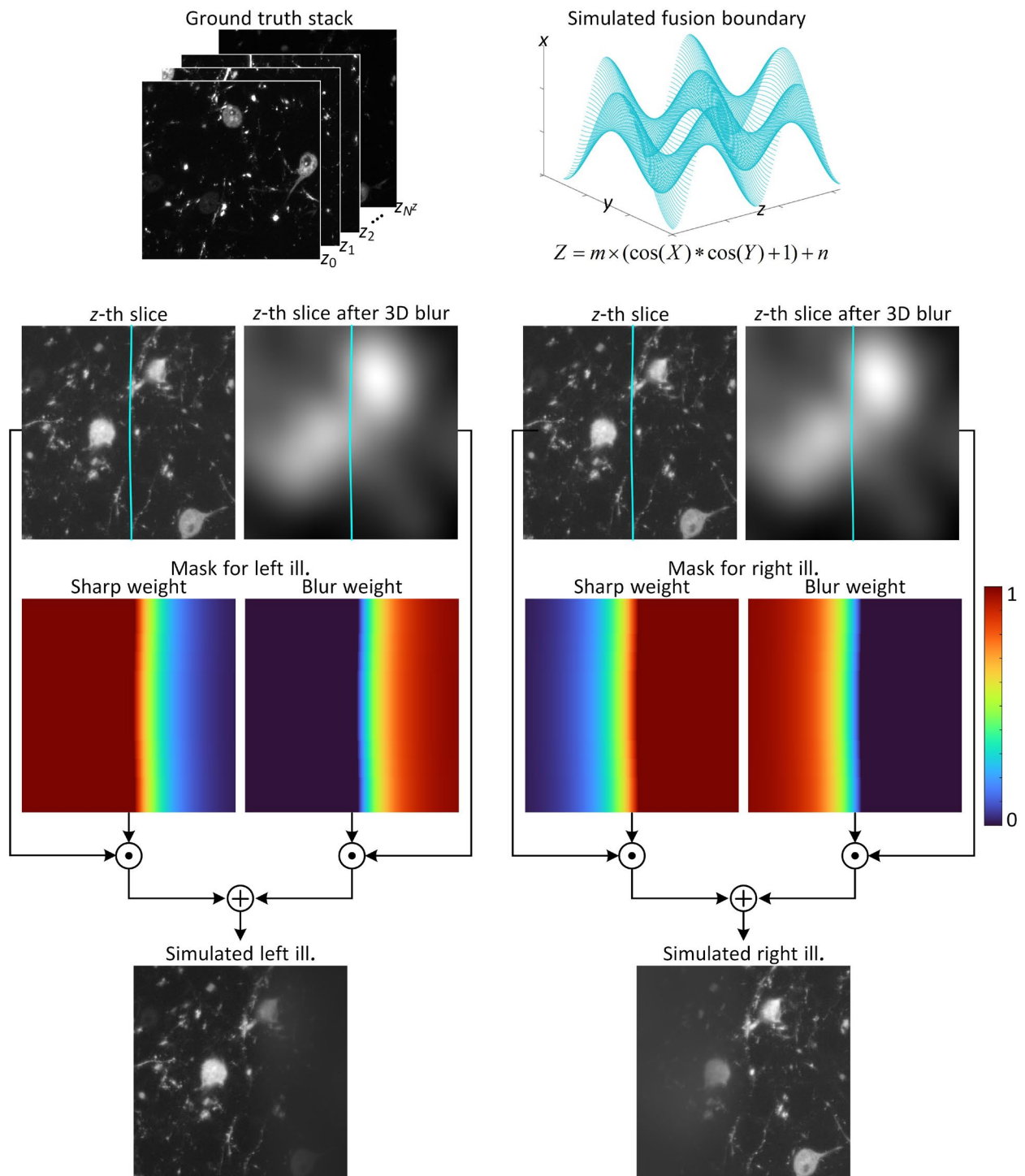
Supplementary Fig. 2. Simulation of striping SPIM. Adipose tissue is used as a stripe-free ground truth image. Structured noise is simulated to mimic horizontal and multiplicative stripes.



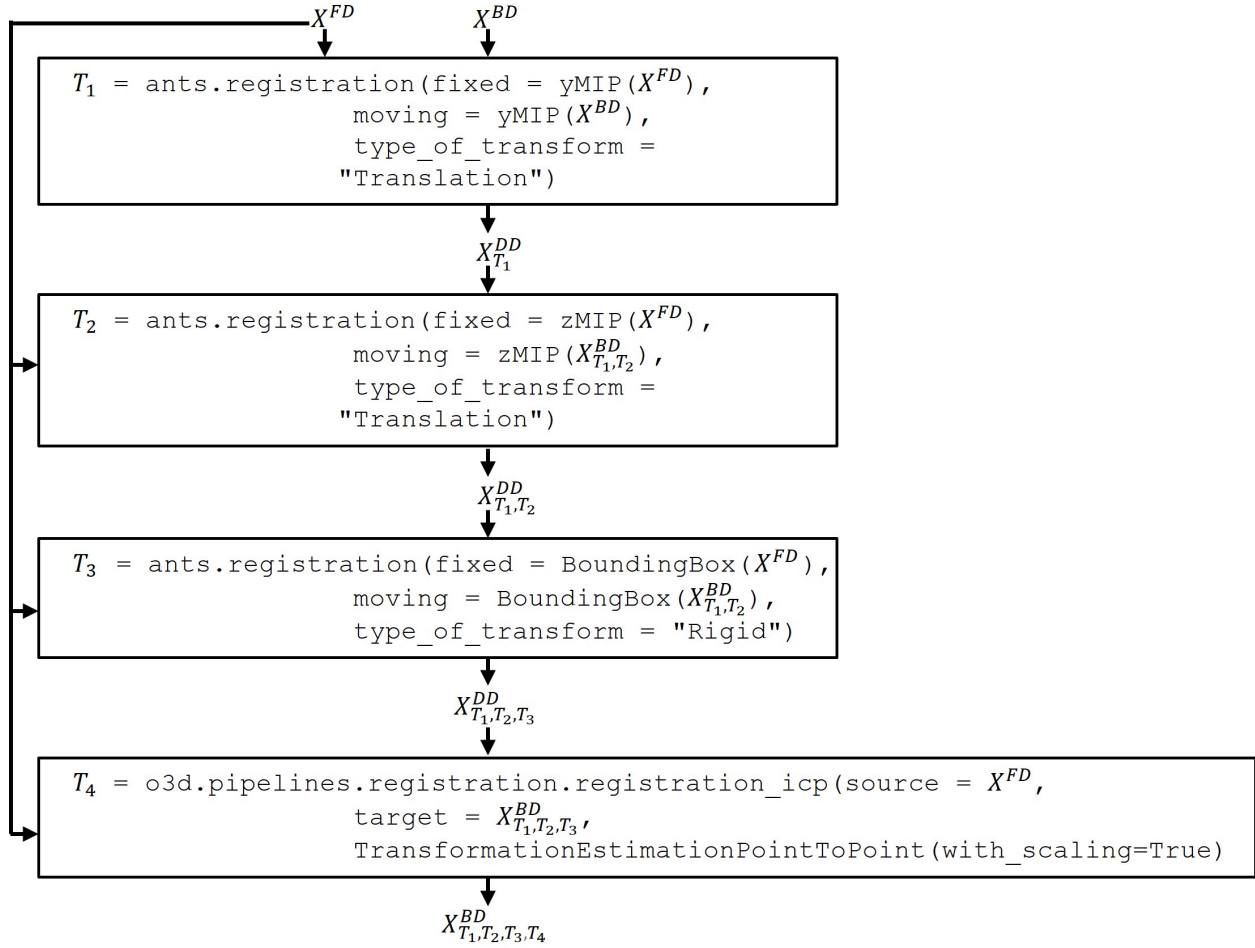
Supplementary Fig. 3. Stripe removal results on zebrafish brain dataset. Leonardo-DeStripe successfully removes the thick stripes without dampening the Fourier projection, compared to the residual stripes from MDSR.



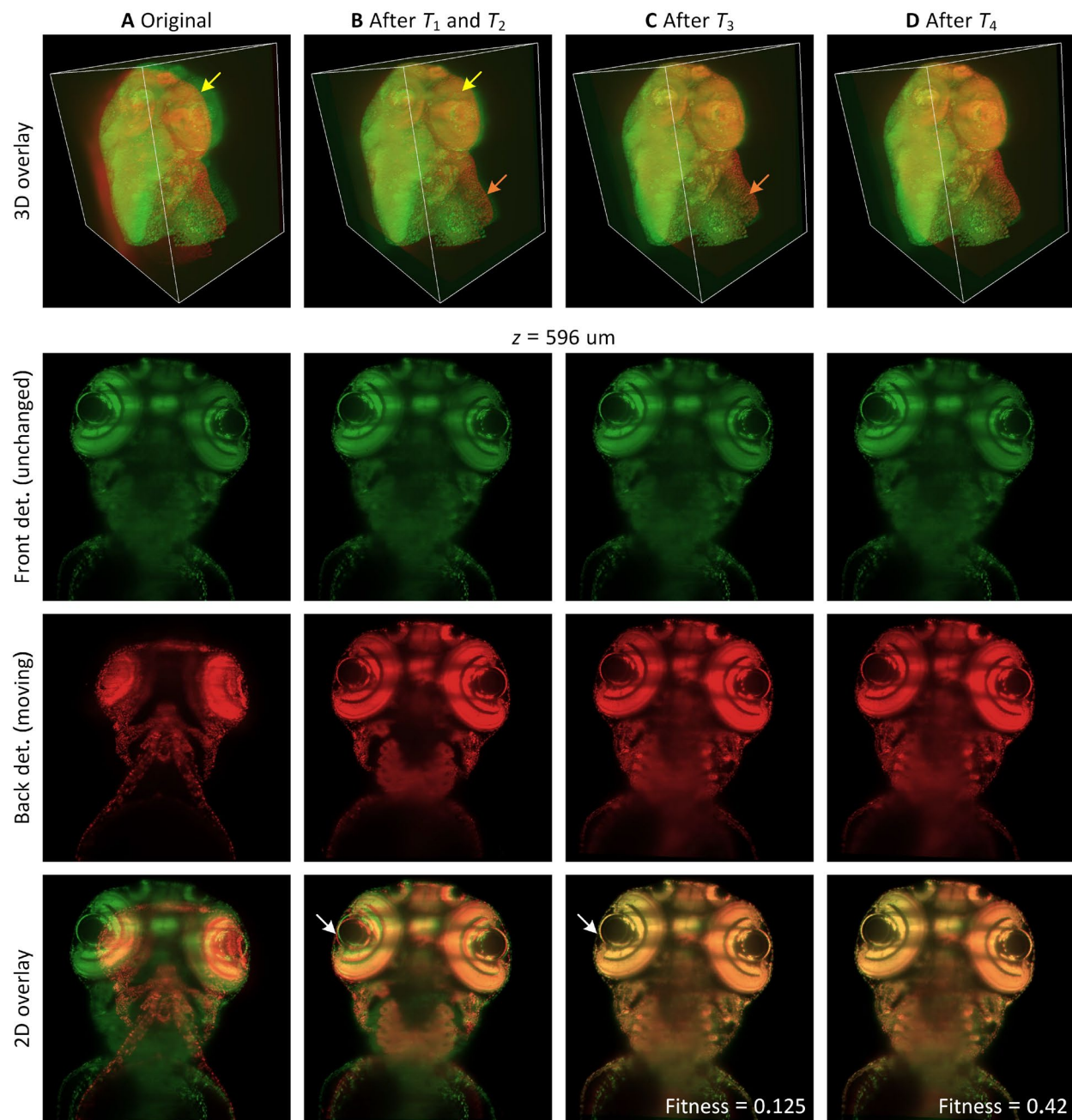
Supplementary Fig. 4. Stripe removal result using Leonardo-DeStripe on a zebrafish brain dataset. The stripes in the input dataset, which are originally horizontal (Fig. 2F in the main text). We manually rotate the stack 45° to test the extensibility of Leonardo-DeStripe.



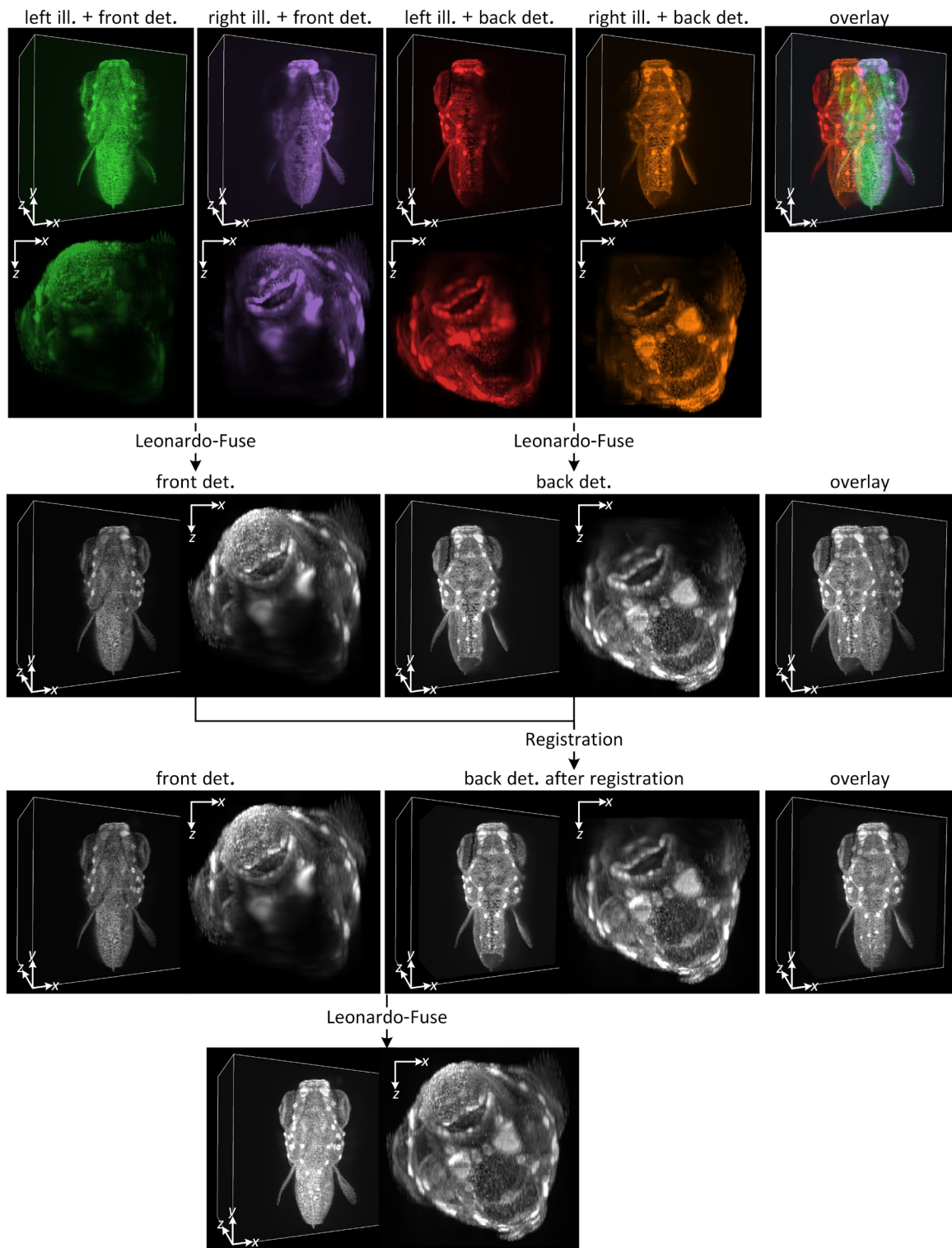
Supplementary Fig. 5. Simulation of two partially degraded SPIM volumes. Deteriorated stacks are simulated to become more degraded as they get farther from the illumination sources.



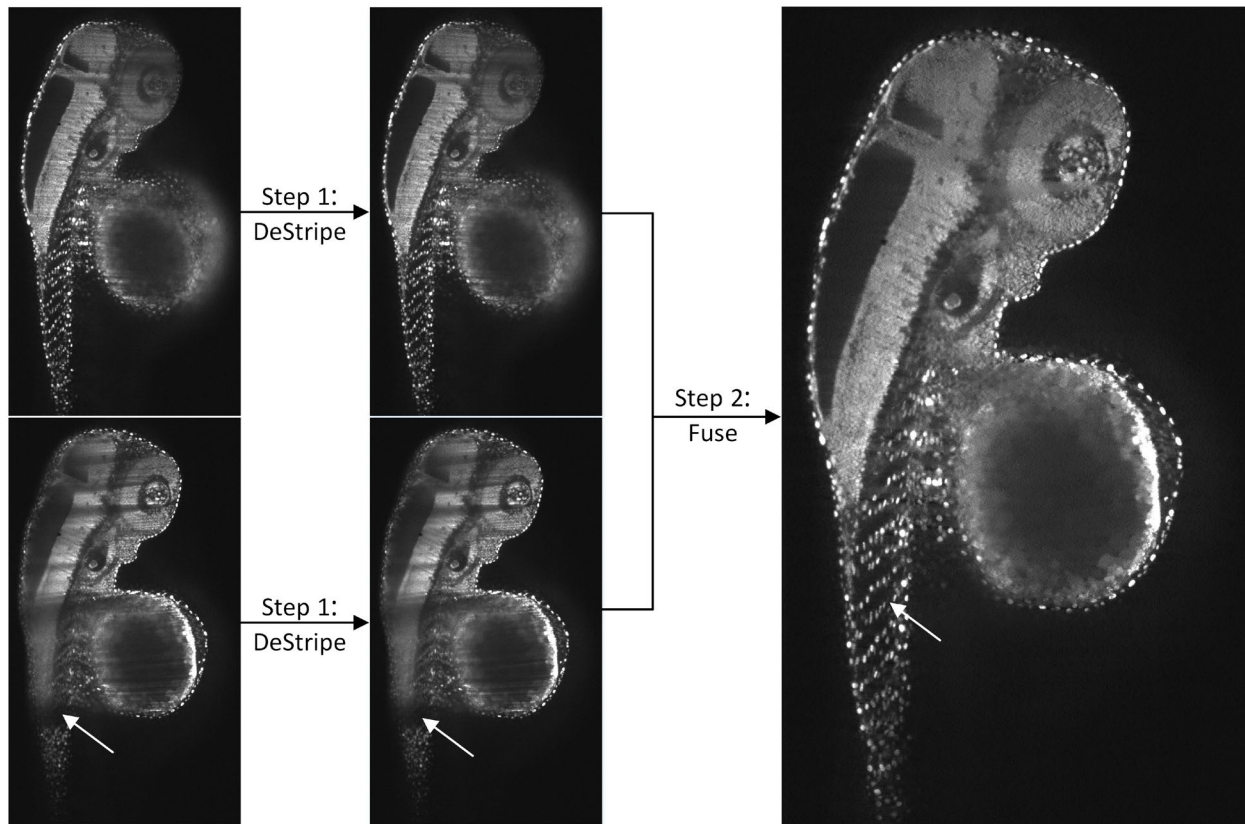
Supplementary Fig. 6. An illustration of registration workflow in Leonardo which is based on ANTsPy (ants). From coarse to fine, Leonardo registers input datasets using their maximum intensity projections (along y axis, i.e., yMIP or along z axis, i.e., zMIP), 3D downsampled and cropped volume stacks (BoundingBox), and extracted point anchors at full resolution, from step to step.



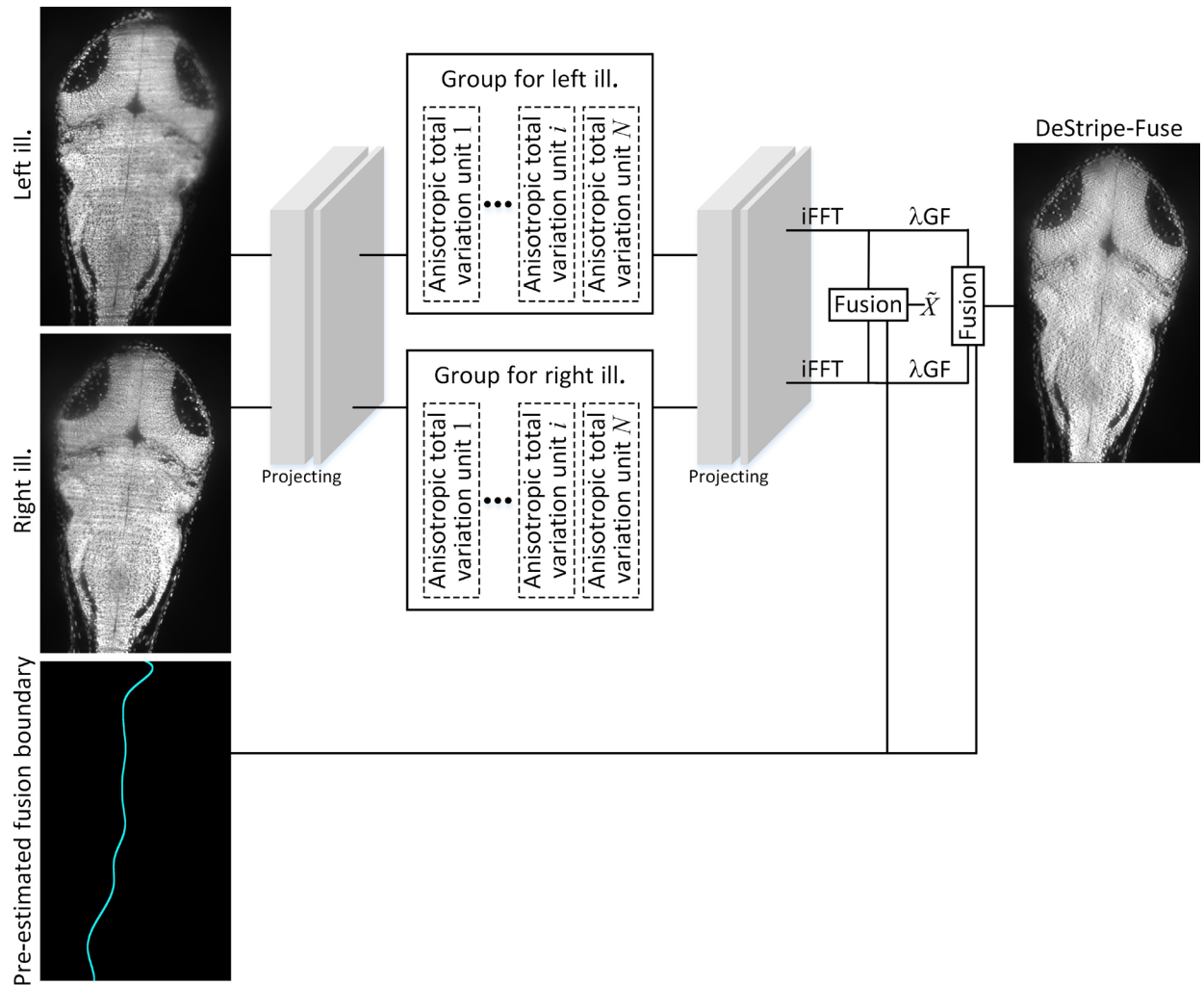
Supplementary Fig. 7. Registration performance from coarse to fine. The stack acquired using camera in the back is gradually aligned with the volume captured with camera in the front (white and orange arrows in the 3D overlay and white arrows in 2D overlay at a depth of 596 μm).



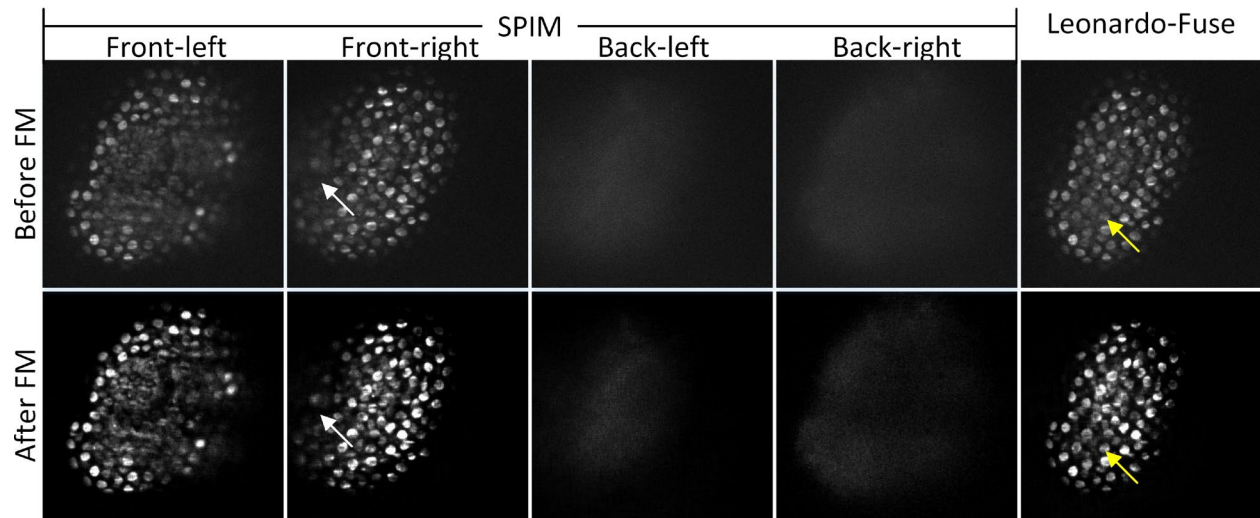
Supplementary Fig. 8. The whole Leonardo-Fuse workflow when registration is required on a H2B-GFP labeled transgenic zebrafish. Leonardo-Fuse (along illumination) is first performed twice for stacks with different detection cameras independently. Registration is then optimized based on fusion result along illumination. Using registered stacks, Leonardo-Fuse (along detection) is realized as final step.



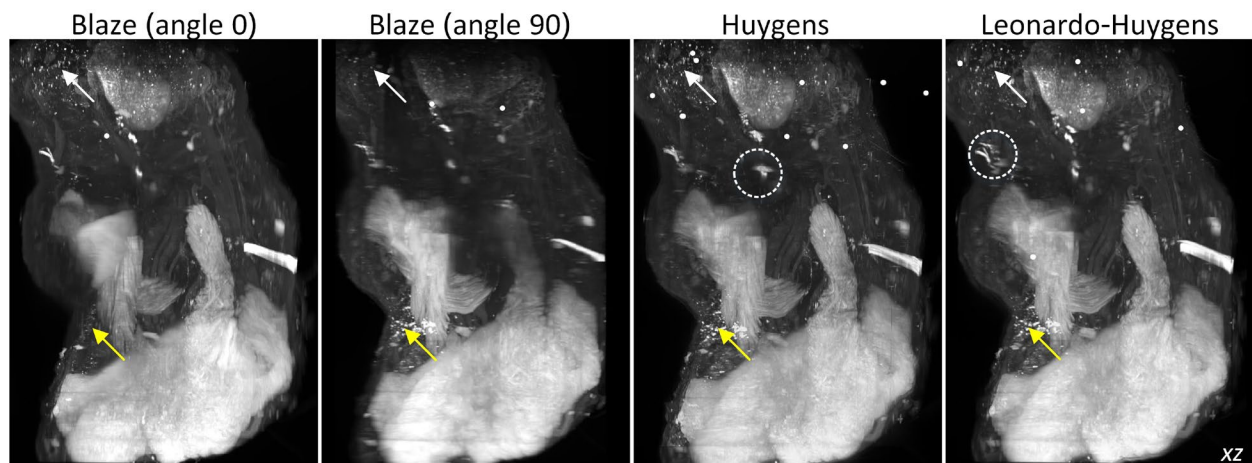
Supplementary Fig. 9. Normal Leonardo workflow, where DeStripe module is performed on individual dataset separately. Fuse module is then used for dataset integration, which is able to not only fuse stack to maximize optical coverage but also resolve remaining extremely thick and dark stripes after DeStripe (white arrows).



Supplementary Fig. 10. Workflow of Leonardo-DeStripe-Fuse. Compared to the normal workflow in which Leonardo-DeStripe is performed multiple times on datasets with opposite illumination orientations independently, this fusion-first approach improves computational efficiency, as Leonardo-DeStripe is performed only once.



Supplementary Fig. 11. Deconvolution results from foundation model UniFMIR, one of the most recent universal fluorescence microscopy-based image restoration models to address different restoration problems. It can be applied to enhance resolution (yellow arrow) right after Leonardo-Fuse removing all blurry regions (white arrow).



Supplementary Fig. 12. Multi-angular fusion results from Huygens. After applying Huygens on fusion results from Leonardo-Fuse, sample information is furthered gathered together (white and yellow arrows). However, misalignment exists (circle regions), as Huygens requires manual registration which is extremely challenging in huge datasets.

Supplementary Notes

Supplementary Note 1: Leonardo-DeStripe

1.1 Stripe remover regularized by anisotropic total variation

Before Leonardo-DeStripe, there were several useful stripe removers which treat the destriping issue as an ill-posed inverse problem. The isotropic total variation (TV) regularized split Bregman framework, which we mentioned in the **Methods** in the main text, is one of the typical:

$$\begin{aligned} &\Leftrightarrow \operatorname{argmin}_{\tilde{X}} \frac{1}{2} \|\tilde{X} - I_s\|_2^2 + \lambda R(\tilde{X}) \\ &\Leftrightarrow \operatorname{argmin}_{\tilde{X}} \frac{1}{2} \|\tilde{X} - I_s\|_2^2 + \lambda \left\{ \|\nabla_x \tilde{X}\|_1 + \|\nabla_y (\tilde{X} - I_s)\|_1 \right\} \end{aligned} \quad (1)$$

where we follow all the definitions that we made in the main text. Since the energy function on \tilde{X} is intractable, split Bregman is adopted to convert the unconstrained minimization problem on \tilde{X} into a constrained one by introducing auxiliary variables $G_x = \nabla_x \tilde{X}$ and $G_y = \nabla_y \tilde{X}$:

$$\operatorname{argmin}_{\tilde{X}} \frac{1}{2} \|\tilde{X} - I_s\|_2^2 + \lambda \left\{ \|G_x\|_1 + \|G_y - \nabla_y I_s\|_1 \right\} \text{ s.t. } G_x = \nabla_x \tilde{X}, G_y = \nabla_y \tilde{X} \quad (2)$$

Subsequently, by using the Bregman iteration to enforce the constraints weakly, Eq. (2) can further be transformed into a non-constrained minimization:

$$\operatorname{argmin}_{\tilde{X}, G_x, G_y} \left\{ \begin{aligned} &+ \frac{1}{2} \|\tilde{X} - I_s\|_2^2 + \lambda \left\{ \|G_x\|_1 + \|G_y - \nabla_y I_s\|_1 \right\} \\ &+ \frac{\alpha}{2} \|G_x - \nabla_x \tilde{X} - B_x\|_2^2 + \frac{\beta}{2} \|G_y - \nabla_y \tilde{X} - B_y\|_2^2 \end{aligned} \right\} \quad (3)$$

here, the minimizations of Eq. (3) with respect to \tilde{X} , G_x and G_y can be decoupled, and thus, they can be further converted into three separate sub-minimization problems:

- the G_x -related subproblem:

$$\operatorname{argmin}_{G_x} \left\{ \lambda \{ \|G_x\|_1 \} + \frac{\alpha}{2} \|G_x - \nabla_x \tilde{X}^{(k)} - B_x^{(k)}\|_2^2 \right\} \quad (4)$$

- the G_y -related subproblem:

$$\operatorname{argmin}_{G_y} \left\{ \lambda \left\{ \|G_y - \nabla_y I_s\|_1 \right\} + \frac{\beta}{2} \|G_y - \nabla_y \tilde{X}^{(k)} - B_y^{(k)}\|_2^2 \right\} \quad (5)$$

where the G_x - and G_y -related subproblems can be solved as in Eq. (6) in the main text by using shrinkage operator.

- the \tilde{X} -related subproblem:

$$\operatorname{argmin}_{\tilde{X}} \left\{ \frac{1}{2} \|\tilde{X} - I_s\|_2^2 + \frac{\alpha}{2} \|G_x^{(k+1)} - \nabla_x \tilde{X} - B_x^{(k)}\|_2^2 + \frac{\beta}{2} \|G_y^{(k+1)} - \nabla_y \tilde{X} - B_y^{(k)}\|_2^2 \right\} \quad (6)$$

which is a least-square problem equivalent to:

$$(I + \alpha \nabla_x^T \nabla_x + \beta \nabla_y^T \nabla_y) \tilde{X} = I_s + \alpha \nabla_x^T (G_x^{k+1} - B_x^k) + \beta \nabla_y^T (G_y^{k+1} - B_y^k) \quad (7)$$

which can be solved by using fast Fourier transform efficiently as in Eq. (6) in the main text.

Additionally, the Bregman variables, $B_x^{(k)}$ and $B_y^{(k)}$ can be updated correspondingly:

$$\begin{cases} B_x^{(k+1)} = B_x^{(k)} + (\nabla_x \tilde{X}^{(k+1)} - G_x^{(k+1)}) \\ B_y^{(k+1)} = B_y^{(k)} + (\nabla_y \tilde{X}^{(k+1)} - G_y^{(k+1)}) \end{cases} \quad (8)$$

We additionally visualize the aforementioned split Bregman during the first iteration as **Figure SN 1.1**. It is clear to see that the deep learning architecture in Leonardo-DeStripe (**Extended Data Fig. 2**) mimics the first iteration in split Bregman optimization, which encourages the interpretability of Leonardo-DeStripe. The only difference is that Leonardo-DeStripe, empowered by the DC branch, is able to ignore $\tilde{X}^{(0)}$, i.e., the stripe-corrupted I_s , when composing $\tilde{X}^{(1)}$, thus only require one-iteration learning when removing the stripes.

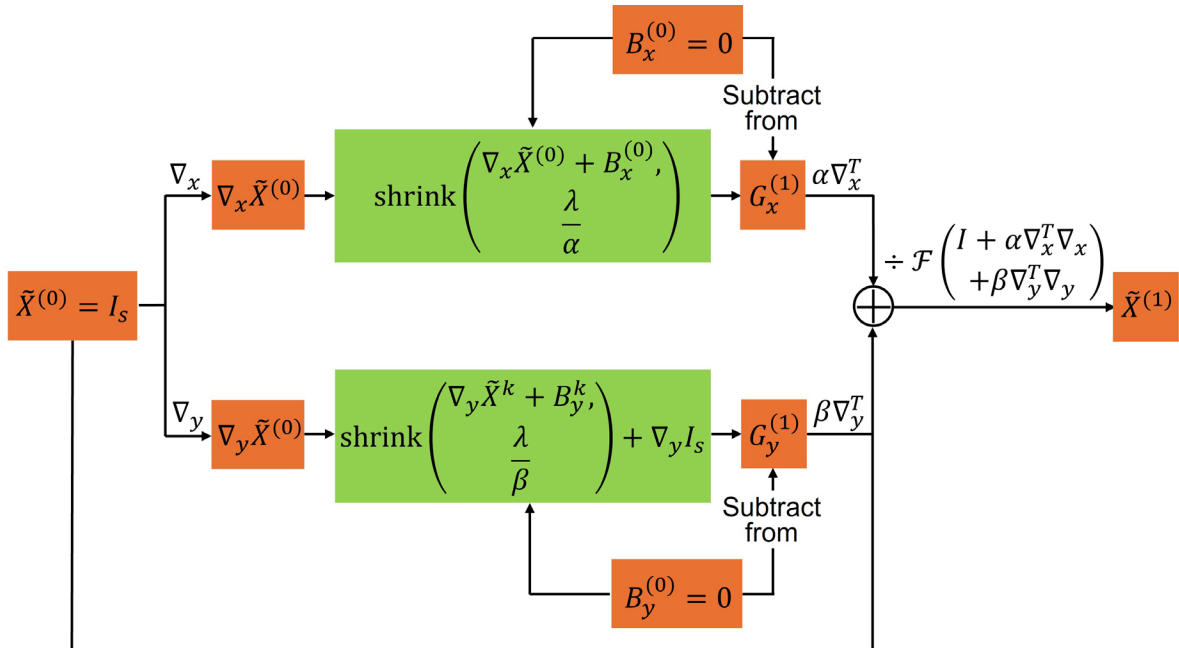


Figure SN 1.1. First iteration of a split Bregman framework regularized with anisotropic total variation.

1.2 Improvement of Leonardo-DeStripe from Bayesian perspective

When analyzing the aforementioned split Bregman framework when being suboptimal in stripe removal, we start by examining Eq. (1) from a Bayesian perspective. Specifically, as restoring a stripe-less \tilde{X} out of the degradation I_s is ill-posed, Eq. (1) originates from the following Maximum A Posteriori (MAP) estimation problem:

$$\begin{aligned}\tilde{X} &= \operatorname{argmax}_X \{ \log(p(I_s|X)) + \log(p(X)) \} \\ \tilde{X} &= \operatorname{argmin}_X \left\{ \frac{1}{2\sigma^2} \|I_s - X\|^2 + R(X) \right\}\end{aligned}\tag{9}$$

where $\log(p(I_s|X))$ is the log-likelihood of observing X given X , log-prior $\log(p(X))$ delivers the prior of stripe-less X and is independent of degraded I_s . Moreover, by assuming pixel-independent Gaussian noise, $\log(p(I_s|X))$ encourages data fidelity between prediction X and input I_s , whereas $\log(p(X))$ becomes anisotropic TV prior $R(X)$. In the other word, trade-off parameter λ in Eq. (1) is equivalent to $2\sigma^2$, that is the standard deviation of the Gaussian distribution we primarily impose on the noise. However, the assumed Gaussian distribution does not hold for noises in stripe removal task, as $I_s - X$ represents the pixel-dependent and non-zero-mean stripe noise. To correct this, $\frac{1}{2\sigma^2} \|I_s - X\|^2$, or σ , should be pixel-variant, which in practice is intractable. Thus, in Leonardo-DeStripe implicitly replaces the $\frac{1}{2\sigma^2} \|I_s - X\|^2$ by:

- solving the G_x -related subproblem using a GNN, where the wedge-shaped mask draws the attention of the stripe resolver to a wedge-shaped region in Fourier perpendicular to the stripe orientation. Conceptionally, threshold $\frac{\lambda}{\alpha} = \frac{2\sigma^2}{\alpha}$ in the shrinkage operator is now spatial-variant with an attention focusing on the stripe-related pixels only.
- composing the stripe-less \tilde{X} based on only G_x and G_y . Hence, $\frac{1}{2\sigma^2} \|I_s - X\|^2$ it totally ignored in the \tilde{X} -related subproblem.

1.3 Guided filtering with various parameters

In Leonardo-DeStripe, guided filtering (GF) has been used multiple times with different setting of size of the window w_k and penalization parameter ϵ . Here, on a murine heart specimen (**Fig. 2E** in the main text), we vary w_k and ϵ , with the striping image serving as both filter input and guidance image, to see the effect. In **Figure SN 1.2**, from left to right, image details of filter output get removed as penalization parameter ϵ getting larger, which is consistent to the previous findings¹. Moreover, due to the directionality of the stripes, w_k , either along column or row, empowers GF with different properties. Specifically, w_k against the stripes, termed column-wise GF in **Figure SN 1.2**, is able to remove the stripes by using larger ϵ , although fine details in the murine heart will be oversmoothed

as well. In comparison, row-wise GF, which operates along the stripes, will not remove the stripes even with $\epsilon = 10$, which, on the other hand, indicates the learnt a_k and b_k being window-based constant, and hence, sample signals will be reserved at most after row-wise GF. This explains why:

- the GF used by Leonardo-DeStripe to refine stripe-less output \tilde{X} of the deep learning should be row-wise and with small ϵ , so as to preserve sample information as much as possible.
- the GF used by Leonardo-DeStripe in the GF-based similarity loss term should be column-wise and with large ϵ , in order to encourage the similarity between learnt stripe-less \tilde{X} and striping I_s after stripes and/or sample details being removed.

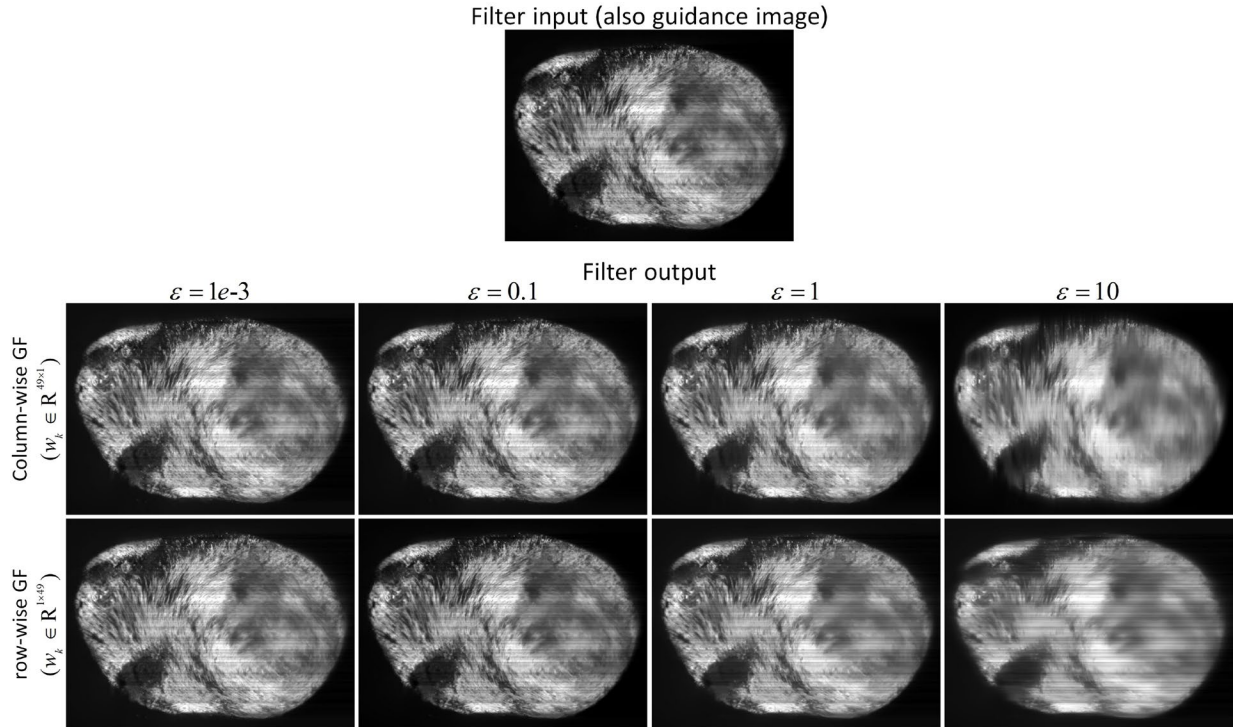


Figure SN 1.2. Column-wise and row-wise GFs with various w_k and ϵ .

1.4 Leonardo-DeStripe is rotatable for stripes with arbitrary directions

Beyond removing horizontal stripes, Leonardo-DeStripe is extensible to resolve stripes with arbitrary angular orientation, as all operations inside are rotatable. Specifically, there are four operators in the Leonardo-DeStripe that need to be rotated when meeting stripes along angle of θ (in radius) (**Extended Data Fig. 2** in the main text):

- the wedge-shaped mask. The orientation of the mask is correspondingly rotated as $\theta + \pi/2$.

- first-order derivative operators including ∇_x , ∇_y , ∇_x^T , and ∇_y^T . In practice, when the orientation of the stripes is along either horizontal or vertical, we use the total variation operator to extract first-order derivative:

$$\nabla_x = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \nabla_y = [-1, 1] \quad (10)$$

whose Fourier projection is given in **Figure SN 1.3A**. Since it's not rotatable, we, instead, use the following alternatives to extract first-order derivative along horizontal and vertical:

$$\nabla_x = \begin{bmatrix} 0.3678 & 0 & -0.3678 \\ 0.6065 & 0 & -0.6065 \\ 0.3678 & 0 & -0.3678 \end{bmatrix}, \quad \nabla_y = \begin{bmatrix} 0.3678 & 0.6065 & 0.3678 \\ 0 & 0 & 0 \\ -0.3678 & -0.6065 & -0.3678 \end{bmatrix} \quad (11)$$

whose Fourier projection is given in **Figure SN 1.3B**. Hence, the derivatives along or against the stripes can be calculated using the following operators, respectively:

$$\begin{aligned} \nabla_{\perp} &= \cos(-\theta\pi)\nabla_x + \sin(-\theta\pi)\nabla_y \\ \nabla_{\parallel} &= \cos\left(-\theta\pi + \frac{\pi}{2}\right)\nabla_x + \sin\left(-\theta\pi + \frac{\pi}{2}\right)\nabla_y \end{aligned} \quad (12)$$

where ∇_{\perp} and ∇_{\parallel} denotes the derivative operator for the direction against or along the stripes, respectively. The Fourier response of ∇_{\perp} and ∇_{\parallel} when $\theta = \pi/4$ are given in **Figure SN 1.3C**. Hence, ∇_x and ∇_y , which are used by the original LeonardoDeStripe, are to be replaced by ∇_{\perp} and ∇_{\parallel} , respectively. ∇_x^T and ∇_y^T can be replaced by the inverse of ∇_{\perp} and ∇_{\parallel} , correspondingly.

- second-order derivative operators including ∇_{xx} , ∇_{xy} and ∇_{yy} . When facing stripes along vertical or horizontal, ∇_{xx} , ∇_{xy} and ∇_{yy} , which are to calculate the second-order derivative of the input images, are defined as Gaussian Hessian kernel:

$$\begin{aligned} \nabla_{xx} &= \frac{x^2 - \sigma^2}{2\pi\sigma^3} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \\ \nabla_{xy} &= \frac{xy}{2\pi\sigma^6} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \\ \nabla_{yy} &= \frac{y^2 - \sigma^2}{2\pi\sigma^3} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right) \end{aligned} \quad (13)$$

where σ is the stand deviation of the Gaussian distribution and can be manually set by the users (1 by default). When digitizing, the kernel size is $2 \times \text{ceil}(6\sigma) + 1$ with $\text{ceil}(\cdot)$ being the ceiling of the input. The Fourier response of Hessian operators when $\sigma = 1$ is given in **Figure SN 1.4A**. When stripes are along degree θ , Hessian operators are rotated correspondingly:

$$\begin{aligned} \nabla_{\perp\perp} &= \text{centerRotate}(\nabla_{xx}, \theta) \\ \nabla_{\perp\parallel} &= \text{centerRotate}(\nabla_{xy}, \theta) \\ \nabla_{\parallel\parallel} &= \text{centerRotate}(\nabla_{yy}, \theta) \end{aligned} \quad (14)$$

where $\text{centerRotate}(X, \theta)$ is to rotate matrix X θ degree with the origin of the rotation being $[\text{ceil}(6\sigma), \text{ceil}(6\sigma)]$. The Fourier response of Hessian operators $\nabla_{\perp\perp}$, $\nabla_{\perp\parallel}$, and $\nabla_{\parallel\parallel}$ when $\sigma = 1$ and $\theta = \pi/3$ is given in **Figure SN 1.4B**.

- window w_k in GFs. the original w_k used by row-wise and column-wise GFs, respectively, are given in **Figure SN 1.5A**. with dealing with stripes oriented θ , w_k should be rotated to be along the stripes, i.e., θ , for row-wise λ GF when refining the output of the neural network. In comparison, the column-wise GF used in the GF-based similarity loss term should be rotated so as to against the stripes, that is $\theta + \pi/2$. The rotation of the window is implemented with `ndimage.rotate` function provided by `Scipy` package in Python. w_k used to resolve stripes with $\theta = \pi/3$ are displayed in **Figure SN 1.5B**.

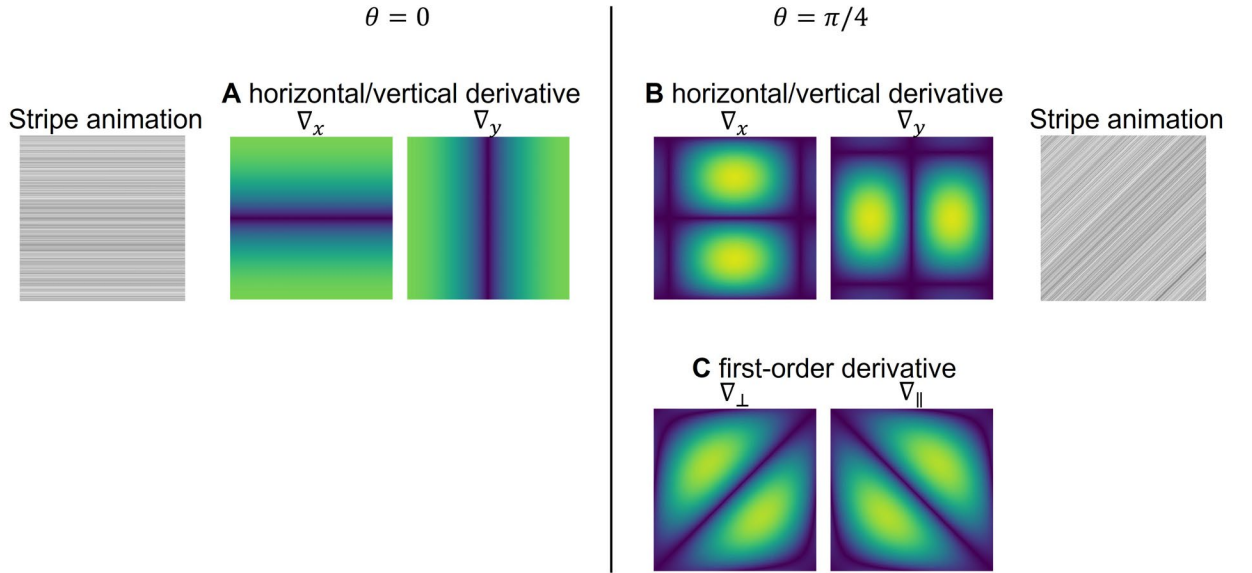


Figure SN 1.3. Fourier response of various derivative operators used by Leonardo-DeStripe in different scenarios.

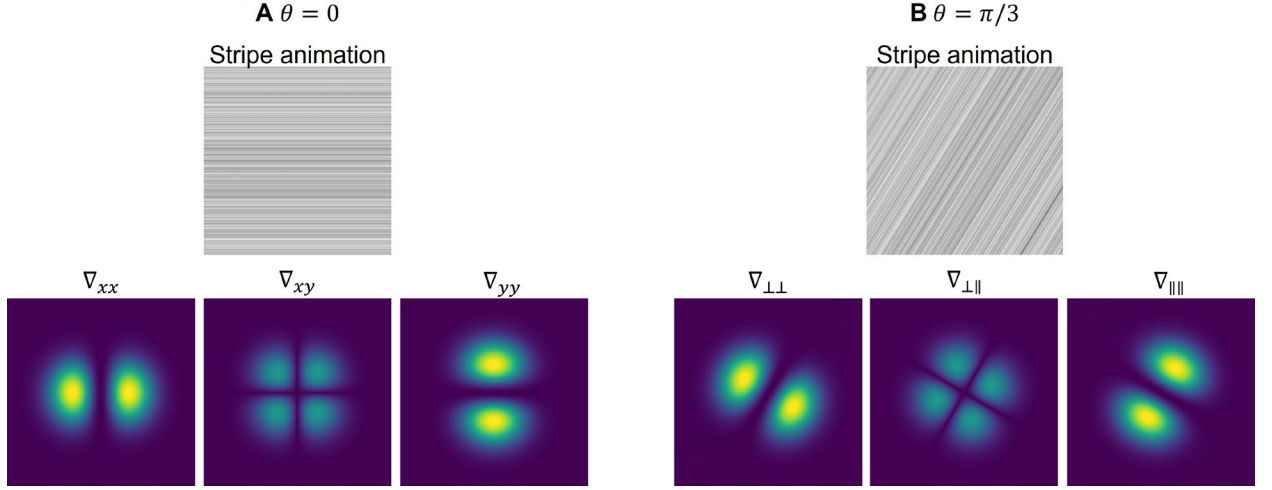


Figure SN 1.4. Fourier response of Hessian operators used by Leonardo-DeStripe in different scenarios.

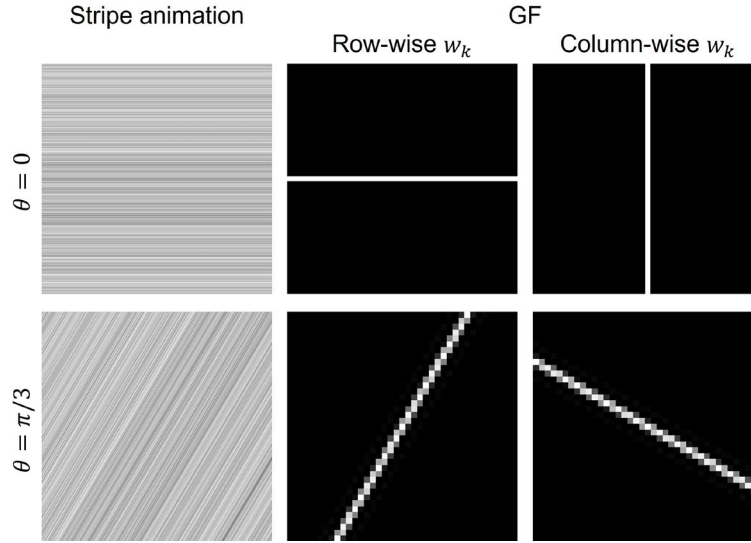


Figure SN 1.5. w_k varies with resolving stripes with different orientations.

Supplementary Note 2: Simulation of striping objects

To quantitatively evaluate the performance of Leonardo-DeStripe, we use the stripe-less stack of the murine heart specimen processed via Leonardo-DeStripe (**Fig. 2E** in the main text) as ground truth and simulate stripe-shape shadows on it (**Supplementary Fig. 2**). Specifically, structured noise can be randomly generated for every slice of the input stack independently. Here we suppose the illumination lens is on the lefthand side. Thus, the structured noise is assumed to be consistent along every row, whose intensity should be

within a range of $[0,1]$ to mimic the absorption of the illumination light. To mimic the sparsity of the absorbing obstacles, the structured noise is randomly sampled from a normal Gaussian distribution with intensities lower than 0.5 reset as 1, i.e., not affected by stripes. Finally, the stripes are blurred using a 2D Gaussian filter with a sigma of 5. The stripe-corrupted SPIM stack can be finally generated by element-wisely multiplying the input stack with the simulated stripes.

Supplementary Note 3: Simulation of SPIM datasets with sequential dual-sided illumination

To quantify the performance of Leonardo-Fuse, specifically -illu branch, in restoring high-quality image stack by fusing the two SPIM datasets illuminated via opposite lenses, we simulate two partially degraded out of a previously published PEGASOS cleared mouse brain labeled with THY1-eGFP ($472 \times 512 \times 512$ voxels)². Since the specimen has been well-cleared in advance, it can be treated as an almost optimal ground truth (GT), that is, image quality is uniformly good as the light sheet penetrating the mouse brain. First, we create a series of pseudo- fusion boundaries

$$\omega_{z,x}^{GT} = 64 \left(\cos \left(\frac{2\pi}{N_x} x - \pi \right) \cos \left(\frac{2\pi}{N_z} z - \pi \right) + 1 \right) + 256 \quad (15)$$

where ω^{GT} means the ground truth fusion boundaries that we simulate, $x \in \{1, 2, \dots, N_x\}$, $z \in \{1, 2, \dots, N_z\}$, for this mouse brain specimen, $N_x = 512$, $N_z = 472$. Thus, $\omega_{z,:}^{GT} \in [192, 320]$ lying in the middle of the z -th slice, is the simulated fusion boundary for the z -th slice. Next, suppose the illumination lens is placed on the lefthand side, we define the part of image on the left side of $\omega_{z,:}^{GT}$ to be unaffected by aberrations, whereas the opposite part is gradually degraded by light scattering along the illumination direction:

$$O_{z,x,y}^{left} = \begin{cases} \tilde{\alpha}_{z,x,y} P_{z,x,y} + (1 - \tilde{\alpha}_{z,x,y}) Q_{z,x,y}, & \text{if } y > \omega_{z,x}^{GT} \\ P_{z,x,y}, & \text{otherwise} \end{cases} \quad (16)$$

where $\tilde{\alpha}_{z,x,y} = \alpha_{z,x,y} / \max_y \alpha_{z,x,y}$ is the weighting parameter at the (z, x, y) position with

$\alpha_{z,x,y} = e^{-5 \frac{|y - \omega_{z,x}^{GT}|}{N_y - \omega_{z,x}^{GT}}}$, O is the simulation result with righthand side degraded as a weighted combination of input sharp volume P and a degraded Q obtained by uniformly blurring P using a 3D Gaussian kernel (sigma of 50). Since $\tilde{\alpha}$ monotonically decrease along the illumination direction, i.e., y -axis, O is simulated to be more affected by the light scattering, i.e., getting blurry, as the illumination light getting deeper into the specimen. Partially degraded $O_{z,x,y}^{right}$ with the illumination lens on the righthand side can be simulated with similar strategy using Eq. (16), with degradation only happens when $y \leq \omega_{z,x}^{GT}$. As a result, the obtained $O_{z,x,y}^{left}$ and $O_{z,x,y}^{right}$ mimic the mouse brain tissue being sequentially illuminated

from the lefthand and righthand sides, respectively. During simulation experiments, results from various fusion algorithms can be quantitatively compared to the ground truth P (**Extended Data Fig. 8** in the main text). Additionally, the fusion boundary learnt by Leonardo-Fuse (along illumination) can also be compared to the pseudo- fusion boundary ω^{GT} .

Supplementary Note 4: Information content assessment

Information content assessment is performed using a combination of discrete cosine transform (DCT-II) and Shannon entropy to measure image quality³. Image patches $I \in \mathbb{R}^{M \times N}$ are firstly transformed into the cosine frequency domain:

$$F_{dct}(u, v) = \frac{2}{N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \delta(i) \delta(j) \cos \left[\frac{\pi u}{2M} (2i + 1) \right] \cos \left[\frac{\pi v}{2N} (2j + 1) \right] I(i, j) \quad (17)$$

$$\text{where } \delta(t) = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } t = 0 \\ 1 & \text{otherwise} \end{cases}$$

where $F_{dct} \in \mathbb{R}^{M \times N}$ is the discrete cosine transform of the patch I . Next, the spectral entropy is used to calculate the information content of the patch I :

$$S_{shannon} = - \sum_{i=1}^n \sum_{j=1}^n p_{i,j} \ln p_{i,j} \quad (18)$$

where $p_{i,j} = \frac{P_{i,j}}{\sum_{i,j} P_{i,j}}$, $P_{i,j} = \frac{F_{dct}(i,j)^2}{(M \times N)^2}$, $S_{shannon}$ is the Shannon entropy of the transformed image.

Supplementary Note 5: SPIM registration

It is fundamentally crucial for fusing SPIM datasets to perform on well-registered SPIM pairs⁴. In most commercial/scientific light sheet-based setups, for example dual-sided illumination lenses in SPIM and Blaze and dual-sided detection lenses in X-SPIM, lenses have been well-aligned in advance. Hence, in Leonardo, there is only one scenario requiring registration by the algorithm, that is when opposite detection lenses are mimicked by rotating the specimen 180°. However, since in different input image stacks, only a complementary part of the specimen can be well-imaged in detail (**Fig. 1B**), registration in SPIM datasets is extremely challenging. Hence, we recommend users of Leonardo-Fuse to manually register the input stacks in advance, or consider bead-based registration plugins⁵ which require embedding fluorescent beads in the mounting medium around the specimen in advance. Nevertheless, in Leonardo-Fuse, we optimize a 3D image-based registration workflow in ANTsPy⁶ which can be useful when pre-registration is not available. Specifically, in order to ensure as much overlap of usable information of the specimen as possible, registration is estimated based on fusion result from Leonardo-

Fuse (along illumination) and then applied to both image stacks captured via back detection lens. Otherwise, for light sheet setups where dual-sided illumination is performed simultaneously, e.g., Blaze, registration is estimated on the input stacks directly. An illustration of the entire workflow of Leonardo-Fuse when registration in-between is required is given in **Supplementary Fig. 6**.

Registration in Leonardo includes three steps in total. Given the two fusion results, one X^{FD} by fusing two stacks with dual-sided illumination and front detection and the other X^{BD} by fusing volumes with dual-sided illumination and back detection, Leonardo firstly translates X^{BD} to the same space as X^{FD} . To fasten the optimization, this translation transformation is learnt based on maximum intensity projection (MIP) in 2D. Specifically, the translation steps along z-axis, namely t_z , and x-axis, termed t_x , are firstly learnt based on MIP of X^{FD} and X^{BD} along y-axis:

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & t_z \\ 0 & 1 & 0 & t_x \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (19)$$

whereas the translation step along y-axis, termed t_y , is learnt based on MIP of X^{FD} and $X_{T_1}^{BD}$ mapped via T_1 along z-axis:

$$T_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (20)$$

as a result, X^{BD} is mapped as X_{T_1, T_2}^{BD} , which is in the same space as X^{FD} via $T_2 \times T_1$.

Next, a Rigid transformation is learnt as fine registration, which includes rotation and translation in 3D simultaneously. As one of the most state-of-the-art toolkits for image registration, ANTsPy houses top-performing algorithms used worldwide by scientific communities for registering biological or medical imaging data. Nevertheless, one of the major drawbacks of ANTsPy is being computationally expensive, especially when processing light sheet-based datasets up to Terabyte scale. Thus, 3D Rigid registration between X^{FD} and X_{T_1, T_2}^{BD} is estimated under 8-bit unsigned integer. Moreover, the registration is estimated within a bounding box. The expansion of the bounding box in xy is defined by segmenting the MIP of X^{FD} and X_{T_1, T_2}^{BD} along z-axis. Meanwhile, its expansion along z is determined such that the bounding box includes no more than $200 \times 1024 \times 1024$ pixels, tested on 50 GB system random-access memory (RAM), in total in 3D. Additionally, in case of system RAM smaller than our testing setup, we allow users to define down-sample ratios, `axial_upsample` and `lateral_upsample` for xy and z, respectively, for registration. As a result, X_{T_1, T_2}^{BD} is mapped as X_{T_1, T_2, T_3}^{BD} after registration using the learnt transformation matrix T_3 in a format of:

$$T_3 = \begin{bmatrix} 1 & m_z & n_z & t_z \\ m_x & 1 & n_x & t_x \\ m_y & n_y & 1 & t_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (21)$$

Finally, in case of the aforementioned Rigid transformation learnt in a down-sampled space, a descriptor-based Affine registration is learnt under the original resolution. Specifically, the affine transformation is estimated based on anchor points extracted using 2D Difference of Gaussian (DoG) method for the sake of memory efficiency. The extracted descriptors are then registered using Iterative Closest Point (ICP) registration algorithm⁷, which is implemented using `Open3D` package in Python.

Overall, registration in Leonardo includes three rounds, that is translation in 2D, Rigid in 3D (potentially down-sampled) and Affine transformation in 3D and full-resolution, from coarse to fine. In addition, lateral and axial resolutions of the input stacks are required by all three rounds of registration.

Supplementary Note 6: Optimized Leonardo-Fuse for large tissue

As a GPU-aided post-processing tool, Leonardo-Fuse requires a decent number of computational resources. Moreover, if registration is needed, sufficient system RAM is crucial (as previously mentioned, ANTsPy, the registration toolbox we use, is computationally demanding). Therefore, when processing extremely large specimens, a specialized version of Leonardo-Fuse has been optimized, as illustrated in **Extended Data Fig. 10A** in the main text. In this version, the input volumes are firstly downsampled, where the sampling ratio can be defined by users based on the capabilities of their computing machine. Especially, this downsampling does not significantly affect the accuracy of the estimated fusion boundary, owing to the continuity of biological specimens. In comparison, the registration matrix learnt in the downsampled space, if estimated, may be less accurate. Therefore, the learned registration matrix is to be refined at full resolution. Considering the significant computational burden of image-based registration, this refinement is realized using only the final round of descriptor-based Affine registration which is described in the previous **Supplementary Note 5**. As a result, given the registered front-back input pair, Leonardo-Fuse can fuse them based on the upsampled fusion boundary. This follows the same strategy to refine the fusion boundary based on GF, as described in **Methods** in the main text.

Supplementary Note 7: Leonardo-DeStripe-Fuse

Given the fusion boundary along illumination, which is pre-estimated on the stripe-corrupted data, and two input slices with opposing orientations, the integration of Leonardo-

DeStripe-Fuse aims to simultaneously remove stripes in both inputs. In this process, Leonardo-DeStripe focuses exclusively on destriping regions that will be incorporated into the final fusion result. Specifically, within the deep learning-parameterized ADMM framework (illustrated in **Extended Data Fig. 2C** in the main text), the two input slices (in the Fourier domain) are first mapped to a high-dimensional feature space using the same MLP^{C} which has been discussed in **Methods** in the main text. In comparison, the subsequent anisotropic TV unit is not shared between the two opposing orientations. Instead, two separate groups of anisotropic TV units are deployed for each input in parallel. Each group may consist of multiple anisotropic TV units, with each unit dedicated to resolving stripes in one specific orientation. Thus, Leonardo-DeStripe-Fuse can be easily extended to multi-directional light sheet-based systems (e.g., UltraMicroscope Blaze). The learned feature maps from both inputs are then projected back to one-dimensional space using the same MLP^{C} . Meanwhile, two DC branches work in parallel to infer the baseline component for each input after stripe correction. The two outputs of the network, denoted as \tilde{X}_1 and \tilde{X}_2 , are finally projected back to the image space using inverse FFT, and later fused into \tilde{X} (following definitions used in **Methods** in the main text) using the same strategy as Leonardo-Fuse (along illumination). The remaining parts, together with the training process, of the network follow the same strategy as Leonardo-DeStripe. The only difference is that when composing full-resolution stripe-less result using λGF , λGF is applied twice separately to \tilde{X}_1 and \tilde{X}_2 , with the outputs fused again based on the fusion boundary.

References

1. He, K., Sun, J. & Tang, X. Guided Image Filtering. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1397–1409 (2013).
2. Dean, K. M. *et al.* Isotropic imaging across spatial scales with axially swept light-sheet microscopy. *Nat. Protoc.* **17**, 2025–2053 (2022).
3. He, J. & Huisken, J. Image quality guided smart rotation improves coverage in microscopy. *Nat. Commun.* **11**, 150 (2020).
4. Li, S., Kang, X. & Hu, J. Image Fusion With Guided Filtering. *IEEE Trans. Image Process.* **22**, 2864–2875 (2013).
5. Hörl, D. *et al.* BigStitcher: reconstructing high-resolution image datasets of cleared and expanded samples. *Nat. Methods* **16**, 870–874 (2019).
6. Tustison, N. J. *et al.* The ANTsX ecosystem for quantitative biological and medical imaging. *Sci. Rep.* **11**, 9068 (2021).
7. Rusinkiewicz, S. & Levoy, M. Efficient variants of the ICP algorithm. in *Proceedings Third International Conference on 3-D Digital Imaging and Modeling* 145–152 (2001). doi:10.1109/IM.2001.924423.