

A: Leonardo-DeStripe napari plugin

Step 1: open the plugin

Step 2: drag-and-drop your datasets

Step 3: select image-to-be-destripe

Step 4: check mandatory parameters

Step 5: check advanced parameters (optional)

Step 6: run

Step 7: save

The interface shows the 'Plugins' menu with 'LSFM Destripe Widget (Leonardo-DeStripe)' selected. The 'Parameters' section includes 'Is vertical' (checked), 'Angle offset(s): 3', 'Upsample kernel length: 49', 'Dark stripe only' (unchecked), 'Mask (label layer): None', and 'Backend: jax'. The 'Advanced parameters' section includes 'Upsample kernel width: 3', 'Downsample ratio: 3', 'Hessian kernel sigma: 1.00', 'Hessian: 1', 'Lambda: TV: 1', and 'MSE: 1'.

B: Leonardo-Fuse napari plugin

Step 1: open the plugin

Step 2: drag-and-drop your datasets

Step 3.1: select fusion type

Step 3.2: select inputs and corresponding illumination directions

Step 4: set path to save intermediate results

Step 5: check mandatory parameters

Step 6: check advanced parameters (optional)

Step 7: run

Step 8: save

The interface shows the 'Parameters' section with 'Require registration' (checked), 'Require flipping along illumination' (checked), 'Require flipping along detection' (checked), 'Keep temporary files' (checked), and 'Camera position (for fuse_illu): front'. The 'Advanced parameters' section includes 'General settings' and 'Registration settings'. The 'Temp path' is set to 'D: \GitHub\lsfm_fusion napari my\intermediates'.

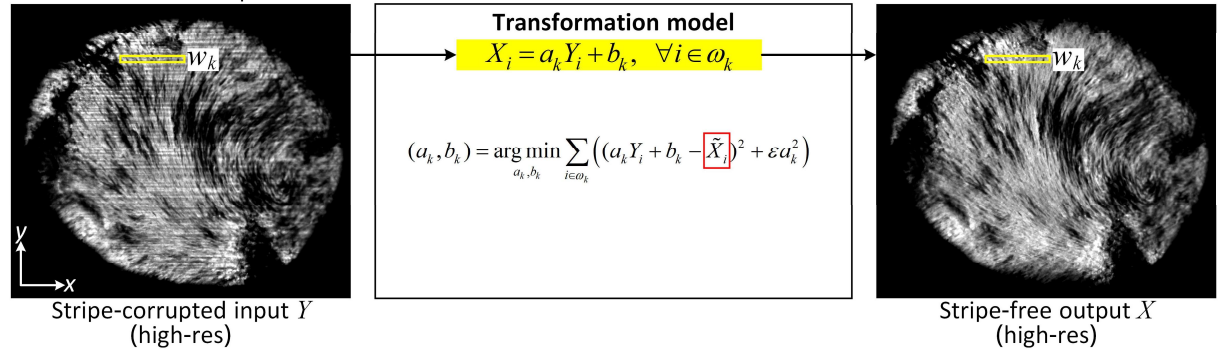
Step 3.1: Fusion type: Select the type of fusion: **Illumination** **Detection**

Step 3.2: Image 1 (camera front): Select the first image from camera front: **D1_I0** **Top**

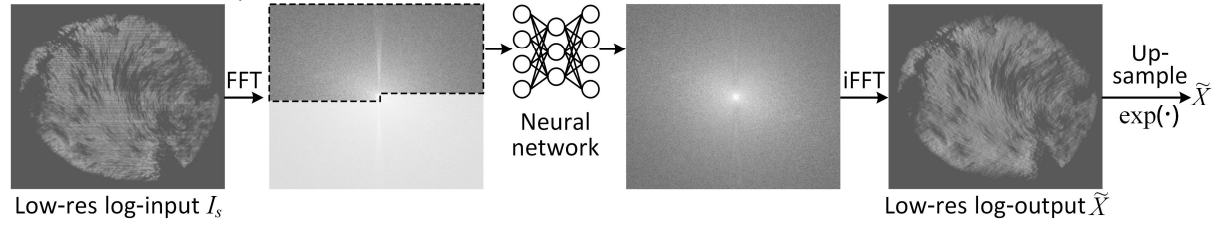
Step 6: General settings: Sparse sample: ☐ Require segmentation: ☒ Lateral downsample: 1 Axial downsample: 1 Downsample in smoothing: 2 Maximum iteration: 50

Extended Data Fig. 1. Graphical user Interface (GUI). Leonardo is modular and capsulized individually in Napari and can adapt to different SPIM variants and workflows. Specifically, Leonardo-DeStripe and Leonardo-Fuse are nested in two Napari widgets separately.

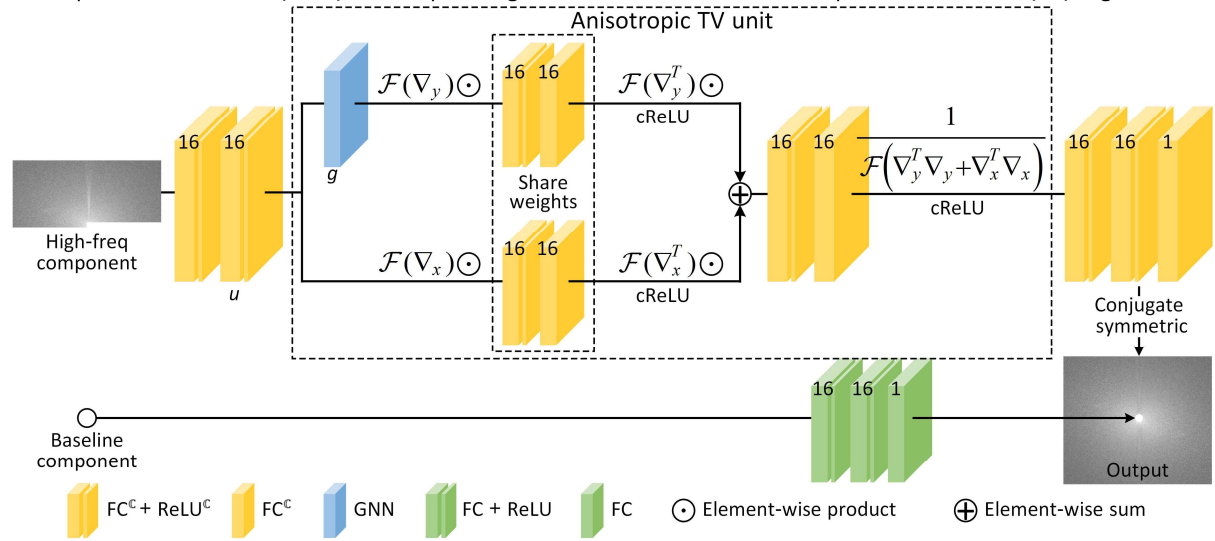
A: Overview of DeStripe



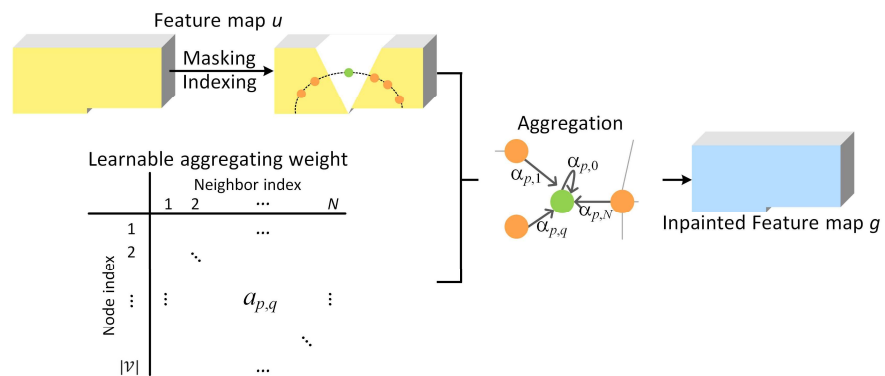
B: Estimation of suboptimal estimation \tilde{X} of X



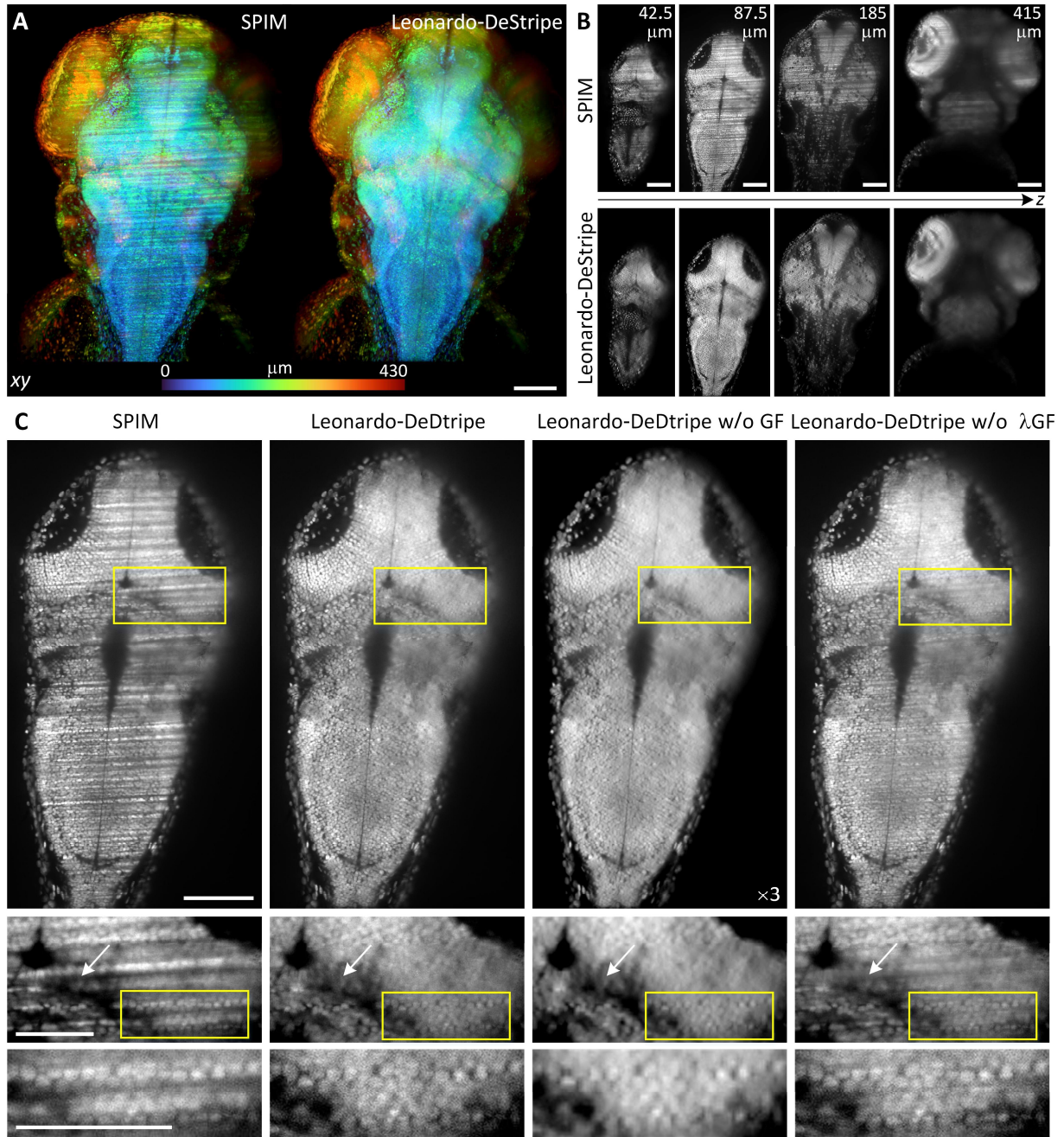
C: Graph neural network (GNN) based split Bregman framework with anisotropic total variation (TV) regularizer



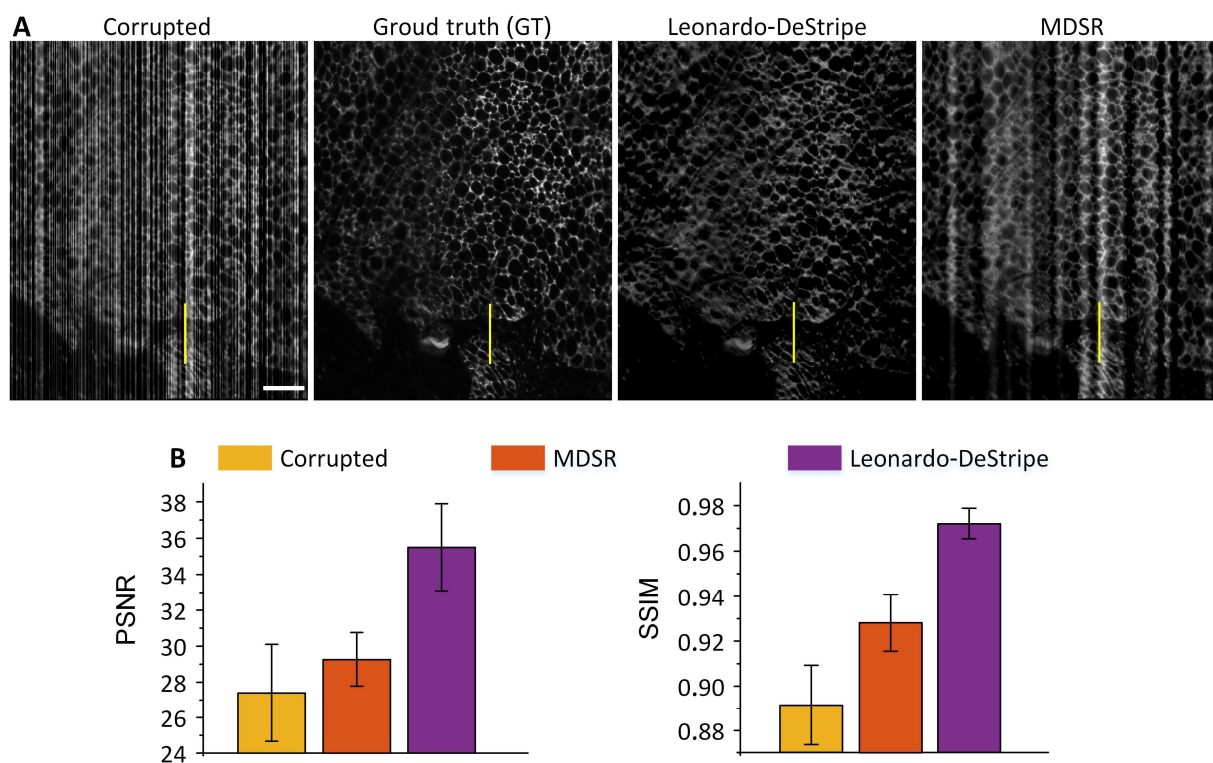
D: GNN



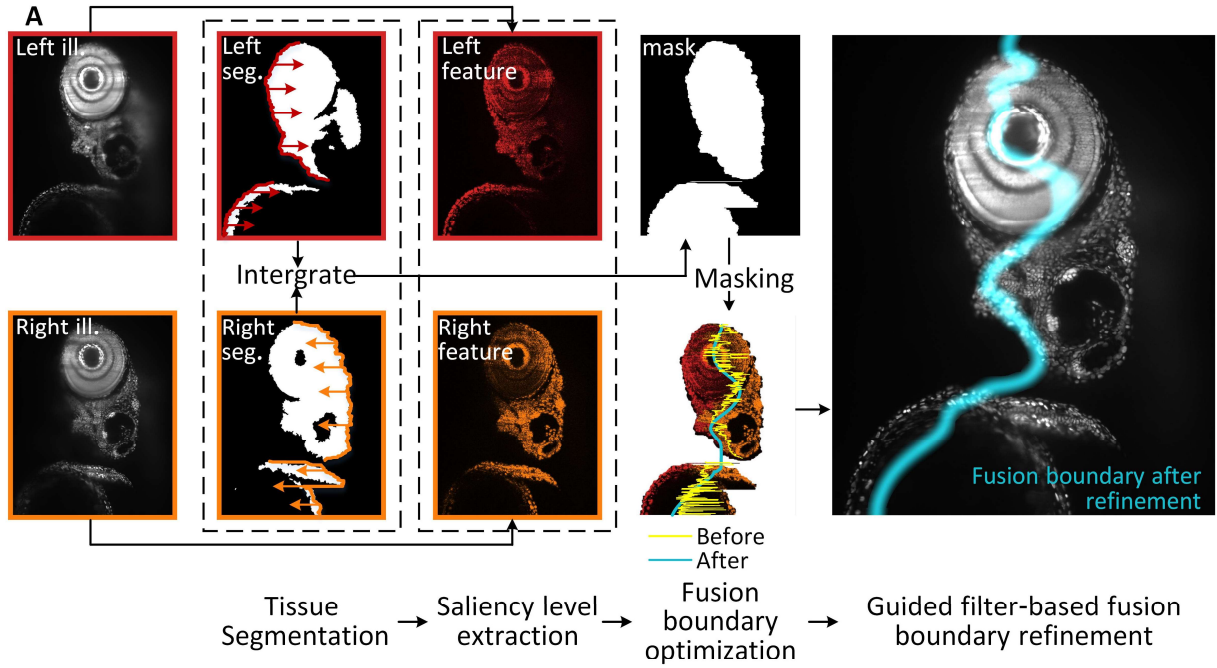
Extended Data Fig. 2. Architecture of Leonardo-DeStripe. (A) Overall, Leonardo-DeStripe treats stripe removal task in SPIM as mapping stripe-corrupted input to stripe-free output via a local linear transformation, supported by the observation that stripes are considered locally consistent along illumination direction. (B) The guidance map used by the local linear transformation, which can be low-resolution and even detail compromised, is learned via a deep learning neural network (NN) in log space after Fourier transformation, where multiplicative stripes are converted into additive noise and thus easier to be modeled. (C) Proposed NN follows a split Bregman framework regularized via anisotropic total variation (TV) in latent feature space. Specifically, a graph neural network in (D) is nested, in which stripe-corrupted Fourier coefficients are inpainted as a combination of their neighbors on a polar coordinate system.



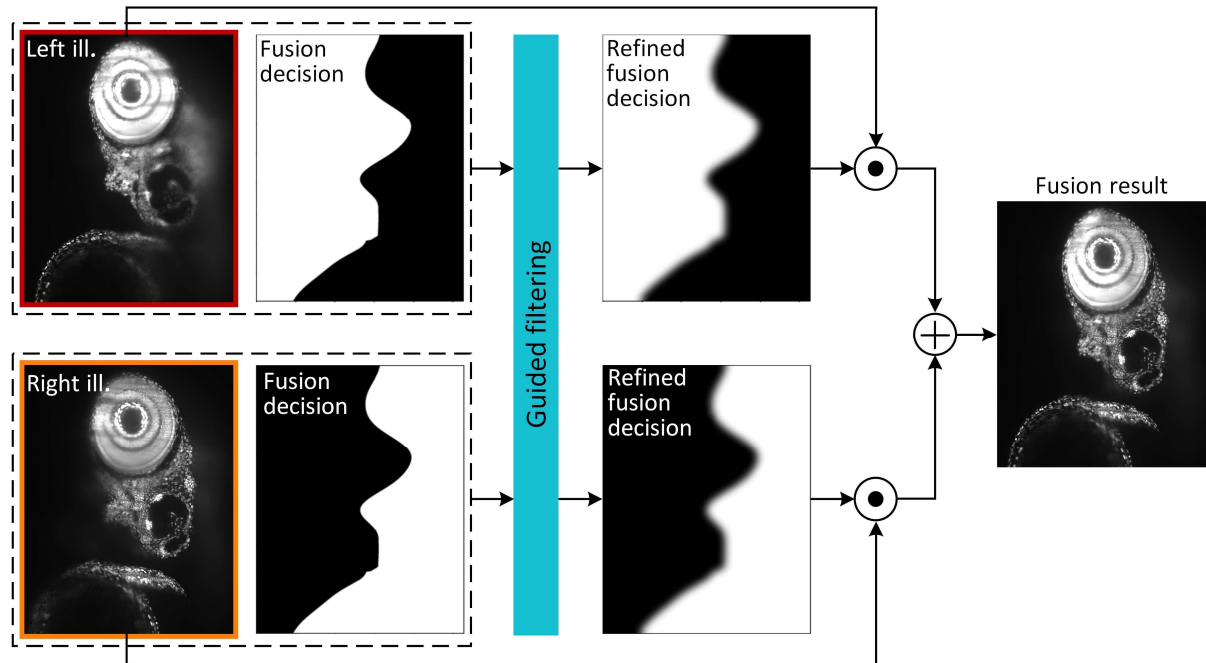
Extended Data Fig. 3. The effect of guided upsampling in Leonardo-DeStripe. (A) Stripes are visible in xy maximum projection, whereas after Leonardo-DeStripe, they are very much resolved. (B) Four typical slices are displayed, where Leonardo-DeStripe resolves diverse stripes successfully. (C) The output of GNN in Leonardo-DeStripe is suitable to serve as guidance map in a color transformation model, although sample details in it may be underscored (Leonardo-DeStripe w/o GF panel). Our modified guided upsampling (termed λGF) works significantly better than conventional guided filtering (Leonardo-DeStripe w/o λGF panel), with stripes being better compressed. Scale bars: 100 μm (A, B, first row in C), 50 μm (last two rows in C).



Extended Data Fig. 4. Simulation results on an adipose tissue. (A) Leonardo-DeStripe' result resembles ground-truth visually, whereas MDSR merges thin stripes into thick ones rather than removing them. (B) Leonardo-DeStripe outperforms benchmarking methods in both PSNR and SSIM. Scale bars: 400 μm (A).

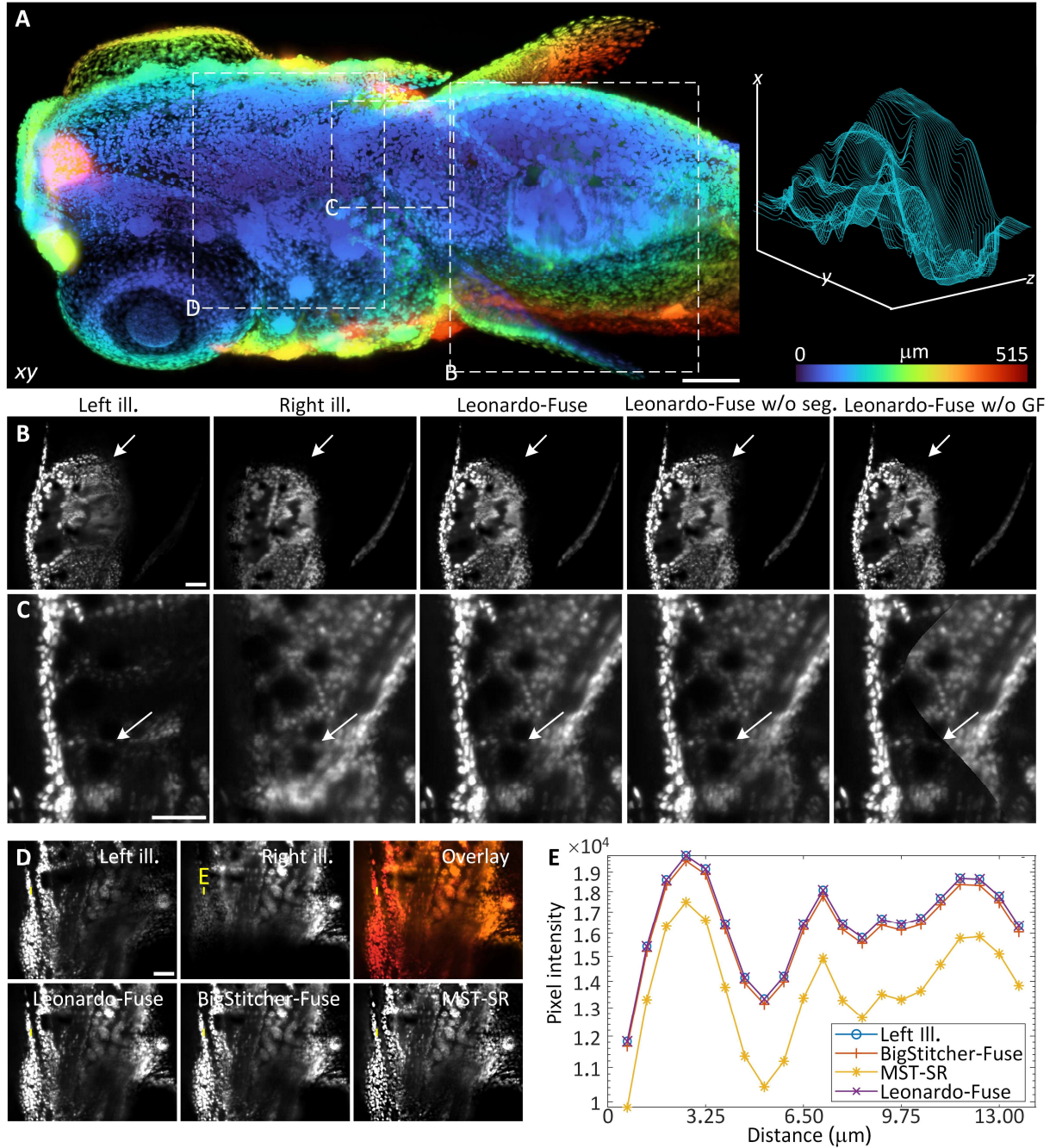


B: guided filtering-based fusion boundary refinement

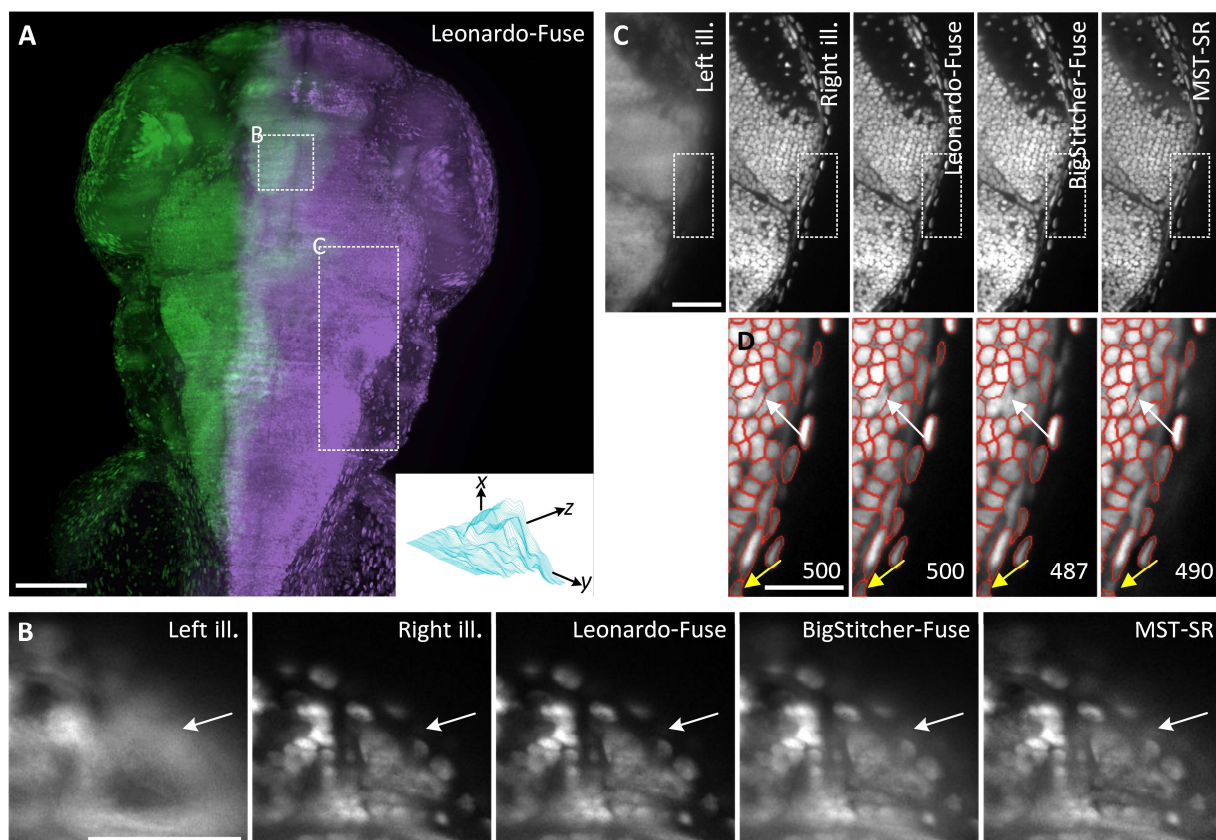


Extended Data Fig. 5. Architecture of Leonardo-Fuse (along illumination). (A) In parallel, Leonardo-Fuse (along illumination) segments sample-related pixels, and meanwhile, quantifies pixel-level image qualities using non-subsampled contourlet transform (NSCT). The used mask to estimate photon propagating path is therefore integrated as regions between left and right boundaries extracted from left seg. and right seg., respectively. Only sample-related saliency levels are then fed into an expectation-maximization algorithm to estimate the fusion boundary, where the smoothness of the fusion boundary, together with the quality of the fused result, are considered alternatively. Before stitching the two SPIM inputs based on the

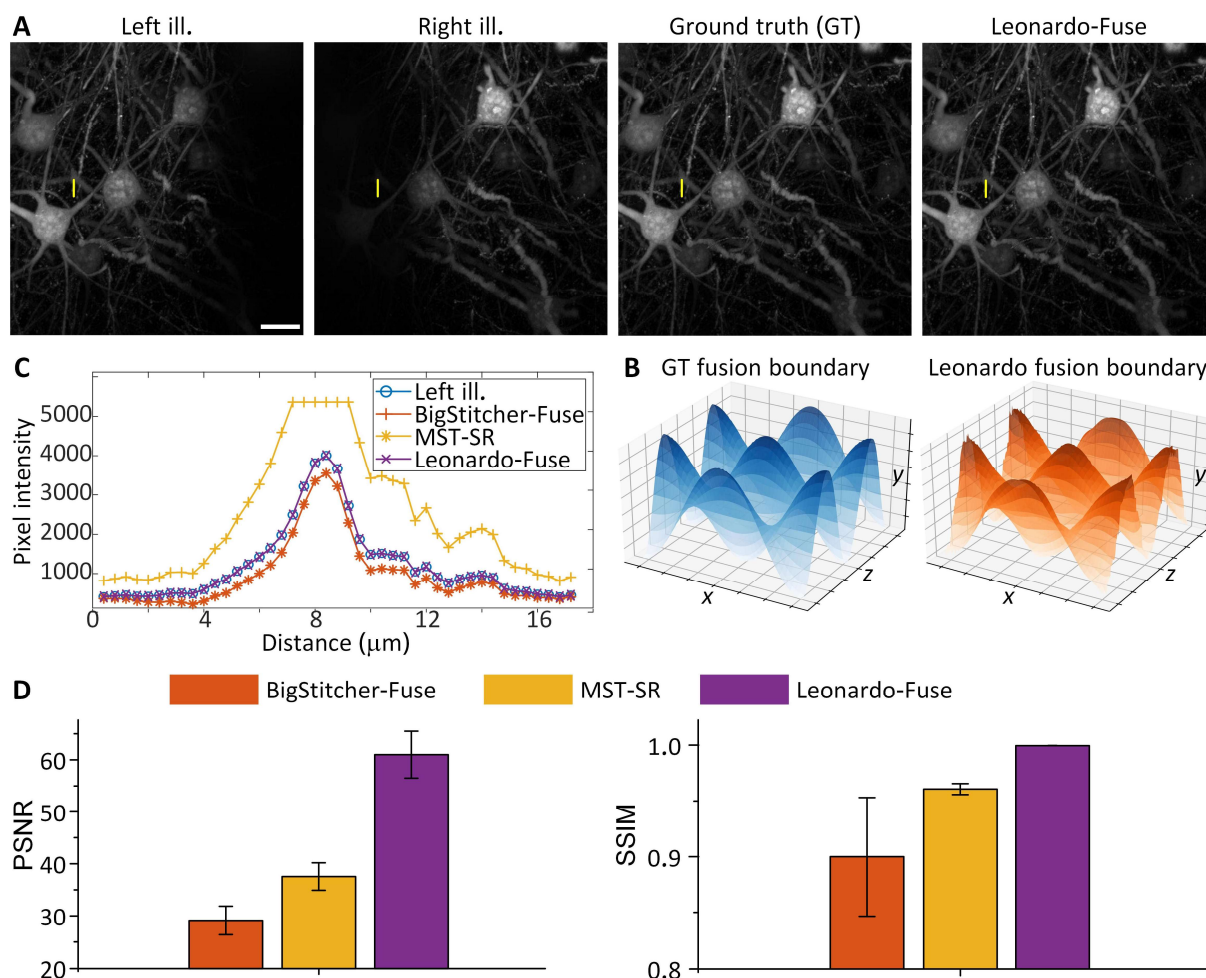
fusion boundary, Leonardo-Fuse (along illumination) additionally refine the fusion boundary using guided filtering (GF) to ensure smooth transitions near the fusion boundary while maintain distinct selections from each stack in areas farther away. The process of refining the fusion boundary using GF is explained in detail in (B).



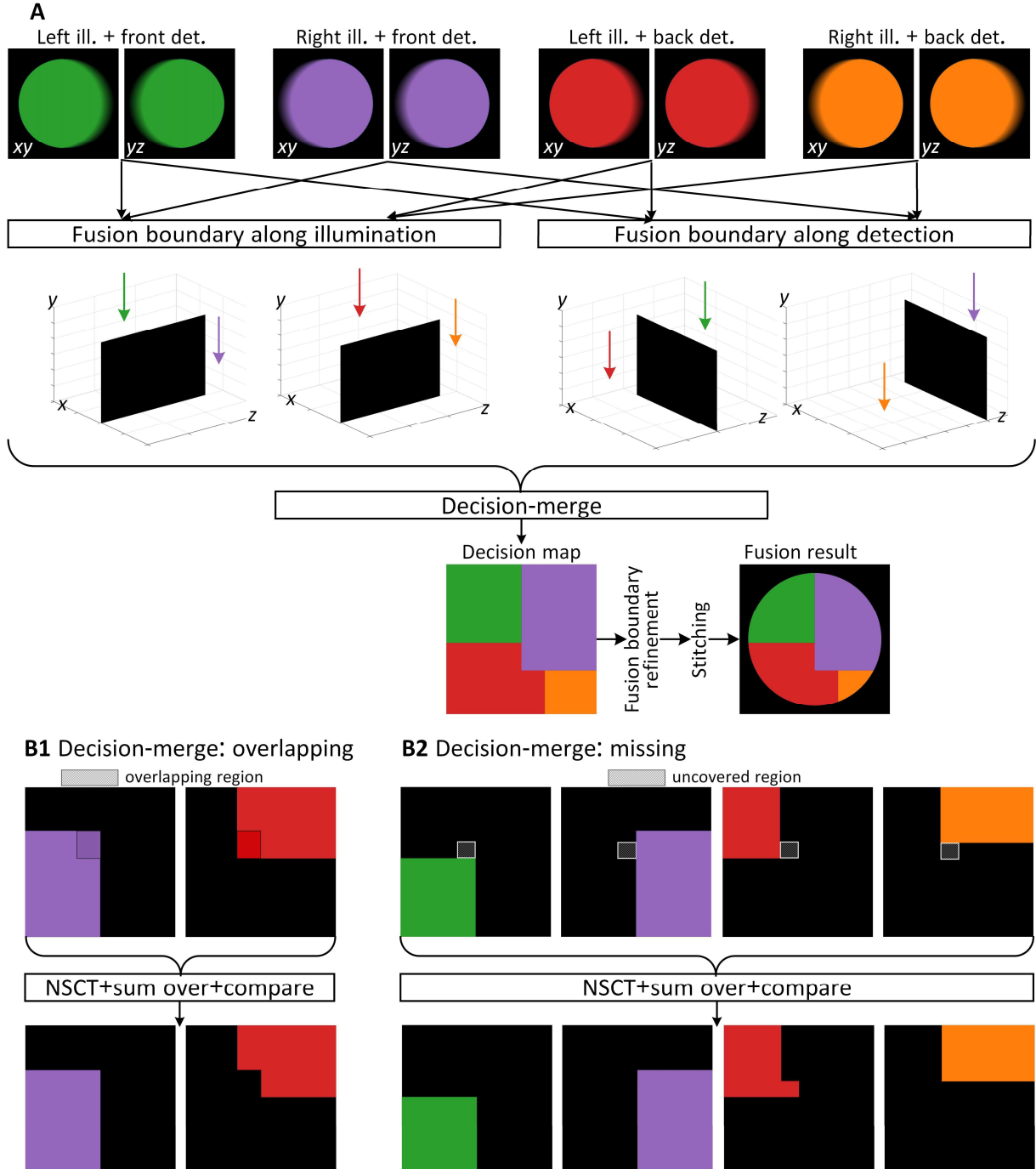
Extended Data Fig. 6. Ablation studies of Leonardo-Fuse (along illumination). (A) Result from Leonardo-Fuse (along illumination) on a H2B-GFP-labeled transgenic zebrafish is given, together with the estimated fusion boundary at various depth plotted on the right. (B) The consideration of prior knowledge of light travel path (enabled by the tissue segmentation) helps Leonardo-Fuse (along illumination) to better reject ghost artifacts (white arrows in Leonardo-Fuse w/o seg. panel). Meanwhile, the guided filtering (GF)-based fusion boundary refinement in (C) allows us to seamlessly stitch multiple datasets without creating artifacts (white arrows in Leonardo-Fuse w/o GF panel). (D) Information is integrated by using both baseline methods and ours. However, we line plot a segment in (E), where only Leonardo-Fuse (along illumination) allows the best data fidelity (perfectly overlap with Left ill.). Scale bars: 100 μm (A), 50 μm (B-D).



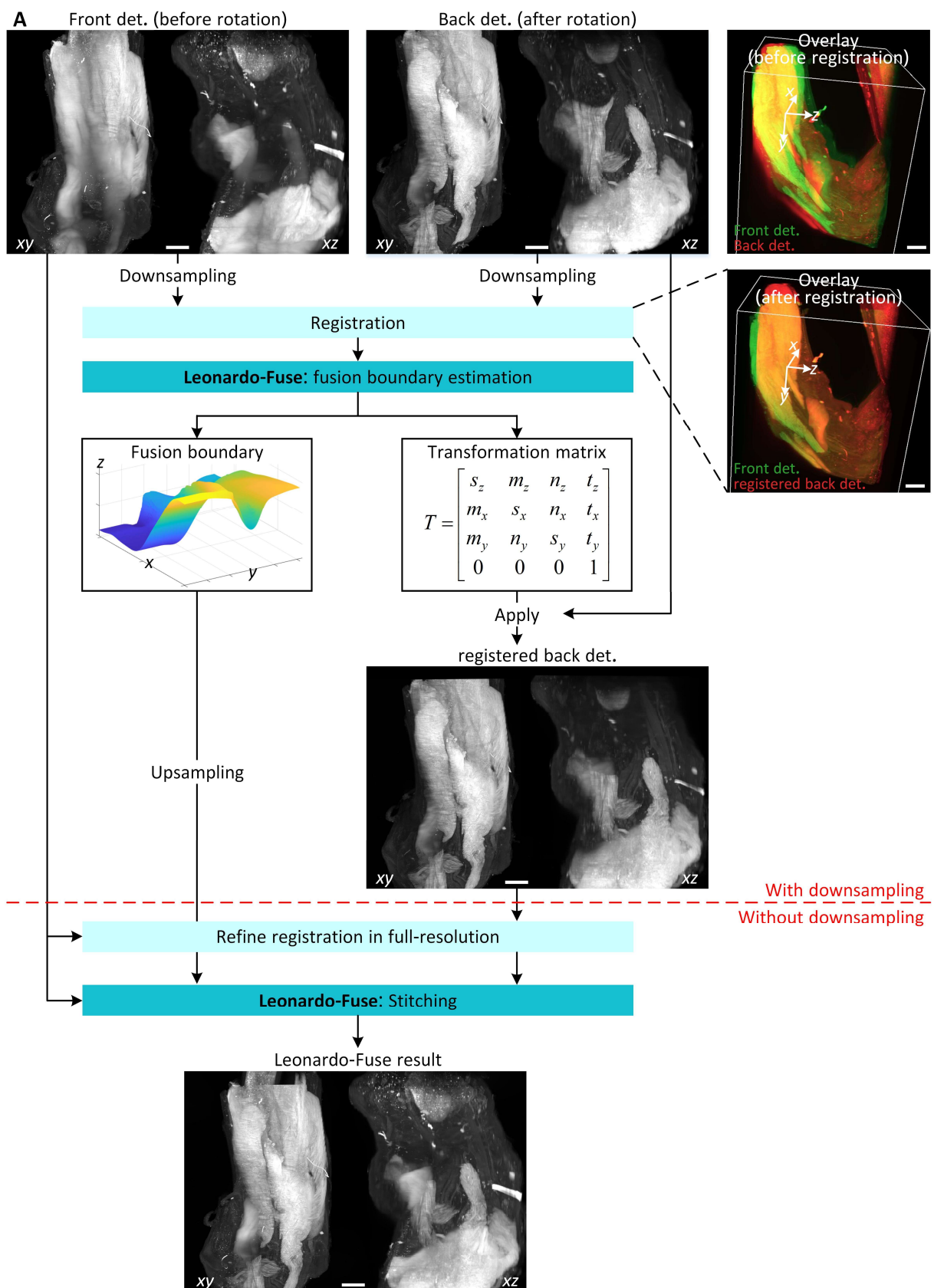
Extended Data Fig. 7. Leonardo-Fuse (along illumination) facilitates downstream segmentation task. (A) Leonardo-Fuse (along illumination) result on calcium imaging specimen is shown, together with the estimated fusion boundary. Two zoom-in regions are given in (B) and (C), where benchmarking methods, especially MST-SR, suffer from halo artifacts (white arrows in (C)). (D) Therefore, when counting cell numbers using Cellpose, only Leonardo-Fuse (along illumination) gives exact same result and thus the best data fidelity as input with illumination source placed on the right. In comparison, baseline approaches fail to detect several cells (white and yellow arrows for BigStitcher-Fuse and MST-SR, respectively). Scale bars: 100 μm (A), 50 μm (B, C), 25 μm (D).

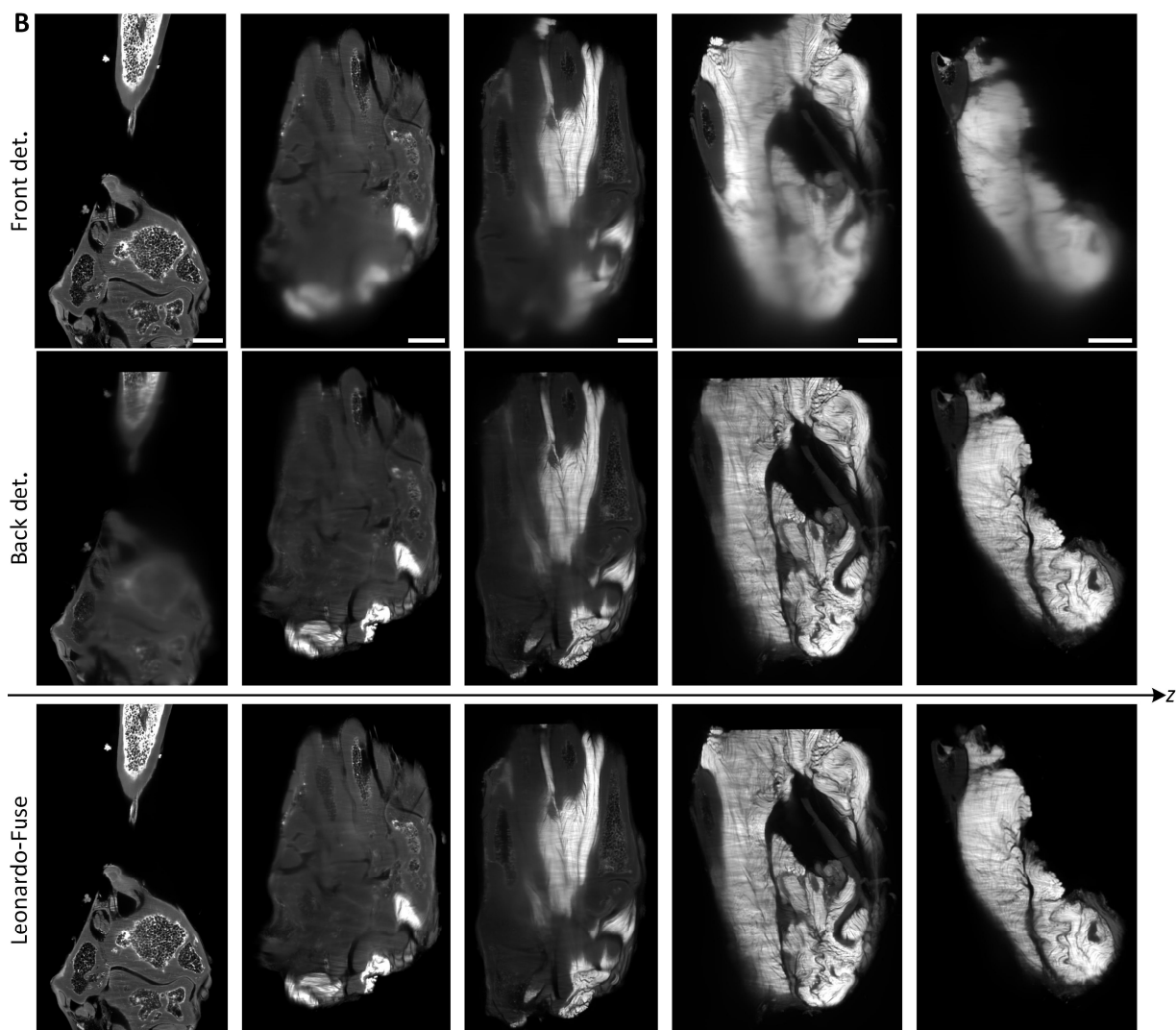


Extended Data Fig. 8. Leonardo-Fuse (along illumination) quantitatively outperforms benchmarking methods. (A) Degradation along illumination is simulated on a previously published PEGASOS-cleared mouse brain. (B) Leonardo-Fuse (along illumination) successfully learns the fusion boundary, very much resemble the ground truth (GT) one. (C) A line plot of a segment (indicated as yellow line in (A)) is given, where only Leonardo-Fuse (along illumination) preserves data fidelity (perfectly overlap with left ill.). (D) Leonardo-Fuse (along illumination) outperforms both benchmarking methods in terms of both PSNR and SSIM. Scale bars: 100 μm (A).



Extended Data Fig. 9 Architecture of Leonardo-Fuse (along detection). (A) Leonardo-Fuse (along detection) reuses the estimation of fusion boundary four times, two of which along illumination direction and the others along detection direction. Since these four “good”-to-“bad” quality boundaries are estimated independently, they need to be merged into one before fed into fusion boundary refinement module and used as guidance for dataset stitching. Decision merge includes two scenarios: (B1) when same region is covered twice and (B2) when a patch is not considered as “good” by any datasets.





Extended Data Fig. 10. Leonardo-Fuse performs fusion on extremely large specimen using manageable computational requirement. (A) Leonardo-Fuse can be optimized to fuse extremely large tissue by estimating the fusion boundary, together with the transformation matrix, in a down-sampled space, and later refining the registration matrix under original resolution, and applying stitching-based fusion technique in high-resolution. (B) Leonardo-Fuse gradually transfers attention from front volume to back stack as getting deeper into the tissue. Scale bars: 500 μm .