nature communications

Article

https://doi.org/10.1038/s41467-023-44354-0

Petascale pipeline for precise alignment of images from serial section electron microscopy

Received: 8 April 2022

Accepted: 7 December 2023

Published online: 04 January 2024

Check for updates

Sergiy Popovych[1,2,5], Thomas Macrina[1,2,5], Nico Kemnitz[1], Manuel Castro[1], Barak Nehoran[1], Zhen Jia[1,2], J. Alexander Bae[1,3], Eric Mitchell[1], Shang Mu[1], Eric T. Trautman[4], Stephan Saalfeld[4], Kai Li[2] & H. Sebastian Seung[1,2]

The reconstruction of neural circuits from serial section electron microscopy (ssEM) images is being accelerated by automatic image segmentation methods. Segmentation accuracy is often limited by the preceding step of aligning 2D section images to create a 3D image stack. Precise and robust alignment in the presence of image artifacts is challenging, especially as datasets are attaining the petascale. We present a computational pipeline for aligning ssEM images with several key elements. Self-supervised convolutional nets are trained via metric learning to encode and align image pairs, and they are used to initialize iterative fine-tuning of alignment. A procedure called vector voting increases robustness to image artifacts or missing image data. For speedup the series is divided into blocks that are distributed to computational workers for alignment. The blocks are aligned to each other by composing transformations with decay, which achieves a global alignment without resorting to a time-consuming global optimization. We apply our pipeline to a whole fly brain dataset, and show improved accuracy relative to prior state of the art. We also demonstrate that our pipeline scales to a cubic millimeter of mouse visual cortex. Our pipeline is publicly available through two open source Python packages.

In serial section electron microscopy (ssEM), a biological sample is sliced into ultra-thin sections, which are collected and imaged at nanoscale resolution[1]. When applied to brain tissue, this technique can yield a 3D image with sufficient resolution for reconstructing neural circuits by tracing neurites and identifying synapses[2]. The technique has been scaled up to image an entire *Drosophila* brain[3], and cubic millimeter volumes of mammalian cortex[4–6].

Purely manual reconstruction of neural circuits from ssEM images is slow and laborious. It is faster to proofread an automated reconstruction generated via convolutional nets[7]. However, automated reconstruction depends on the preceding step of aligning the 2D section images into a 3D image stack. Misalignments can be the dominant cause of errors in the automated reconstructions, so it is important for the alignment to be precise and robust to image artifacts.

State-of-the-art alignment methods identify a sparse set of corresponding points between neighboring images, and use an offline solver to find image transformations that register corresponding points subject to an elastic regularizer that prevents nonsmooth transformations[8–12]. Correspondences may be identified by matching image patches through some kind of cross-correlation[9,11], or through matching hand-designed features such as SIFT[13].

[1]Princeton Neuroscience Institute, Princeton University, Princeton, NJ, USA. [2]Computer Science Department, Princeton University, Princeton, NJ, USA. [3]Electrical and Computer Engineering Department, Princeton University, Princeton, NJ, USA. [4]HHMI Janelia Research Campus, Ashburn, VA, USA. [5]These authors contributed equally: Sergiy Popovych, Thomas Macrina. e-mail: sseung@princeton.edu

While these state-of-the-art methods are often sufficient to coarsely align images, they are inadequate in a number of ways to precisely align. First, cracks and folds are common defects in serial sections (Fig. 1a). Correctly aligning pixels on both sides of a crack or fold (Fig. 1b,c) requires moving closely neighboring pixels in opposite directions. Such a nonsmooth transformation is not possible by interpolating between sparse correspondences, as done by state-of-the-art methods. And even if the density of correspondences is increased, the elastic regularizer also prevents nonsmooth transformations.

Second, alignment must robustly handle image artifacts. Consider a case where striped artifacts caused by knife chatter (Fig. 1d, top left) occur in two sections in a row. High similarity alignment between such a section pair can be produced in two ways: the correct alignment that aligns biological objects, and an erroneous alignment that aligns the stripe patterns. General purpose image feature extractors, such as SIFT[13], do not prevent such errors. While there exist methods that are able to effectively suppress EM image artifacts *after* the dataset has been aligned[14], robust artifact suppression in EM images prior to alignment has not been demonstrated.

Third, even without image artifacts, state-of-the-art approaches make misalignment errors due to false correspondences. The frequency of errors is reduced by hand-designed heuristics for rejecting false correspondences[9,10,12]. For a large dataset, it is difficult or impossible to set the parameters of the heuristics to remove all false correspondences, so manual intervention at the level of single sections or even single correspondences becomes necessary[15].

Fourth, the offline solver in state-of-the-art methods does not scale well to large datasets. The number of variables in the optimization increases with image volume and the complexity of the alignment transformations. We have found that the optimization can be slow because long-wavelength modes of an elastic energy tend to relax slowly. It could be possible to speed up convergence by advanced optimization techniques. Here we take a different approach, which is to completely eliminate the need for a global optimization over the entire dataset.

The above difficulties are all overcome by our computational pipeline for precise alignment of ssEM images. The input to the pipeline is assumed to have been already aligned coarsely, which can be done by conventional methods. The global rotations and scale changes in the input, for example, should be small. We show that our pipeline significantly improves alignment over state-of-the-art methods on the challenging whole brain female adult fly dataset (FAFB) and can align a petascale mouse cortex dataset[5]. The pipeline demonstrates ability to align near cracks and folds up to 10 μm wide.

Our pipeline contains several elements. First, we use metric learning to train convolutional nets to extract encodings of EM images (Fig. 6), and align the images using the encodings (Fig. 5). The method of training is called Siamese Encoding and Alignment by Multiscale Learning with Self Supervision (SEAMLeSS), though there are some differences from a preliminary version of the method[16] that will be described later. Alignment based on the encodings is more robust, because image artifacts are suppressed by the encodings. The convolutional nets are applied in a coarse-to-fine hierarchy (Fig. 2a), which enables correction of large displacements. The use of convolutional nets to compute saturated correspondences between images is now standard in computer vision[17–22]. Our approach extends prior work to correct nonsmooth distortions and large displacements.

Second, the alignment transformation is fine-tuned by gradient descent on a cost function that is the squared difference between the image encodings plus an elastic regularizer. Gradient descent is initialized at the alignment transformation that was generated by the convolutional net. The fine-tuning achieves more precise alignment than SEAMLeSS alone, at the expense of moderately more computation. Both in SEAMLeSS training and in fine-tuning, the elastic regularizer is ignored at locations where nonsmooth transformations are required, using a convolutional net trained to detect cracks and folds[23].

Third, for more robust alignment, each section is aligned to several preceding sections and a procedure called vector voting is used to arrive at a consensus alignment. If the data were defect-free, it would be sufficient to align each section to the previous section. But virtually every section contains one or more regions with defects or missing data. Therefore it is helpful to also align to sections that are further back in the series. Previous elastic alignment schemes added springs between the next nearest neighbor and further sections, motivated by similar considerations[9]. The advantage of vector voting over naive averaging schemes is that the effect of outliers on final alignment is minimized (Fig. 3a).

Fourth, we divide the entire series of sections into blocks of contiguous sections, and sequentially align the sections in each block. The blocks are distributed over multiple computational workers, speeding up the computation.

Fifth, we have to combine the aligned blocks into a single global alignment. Naively, this could be done by aligning the start of each block with the end of the previous block, and then composing these transformations. If there are many blocks, however, the composition of many non-rigid transformations could result in highly distorted

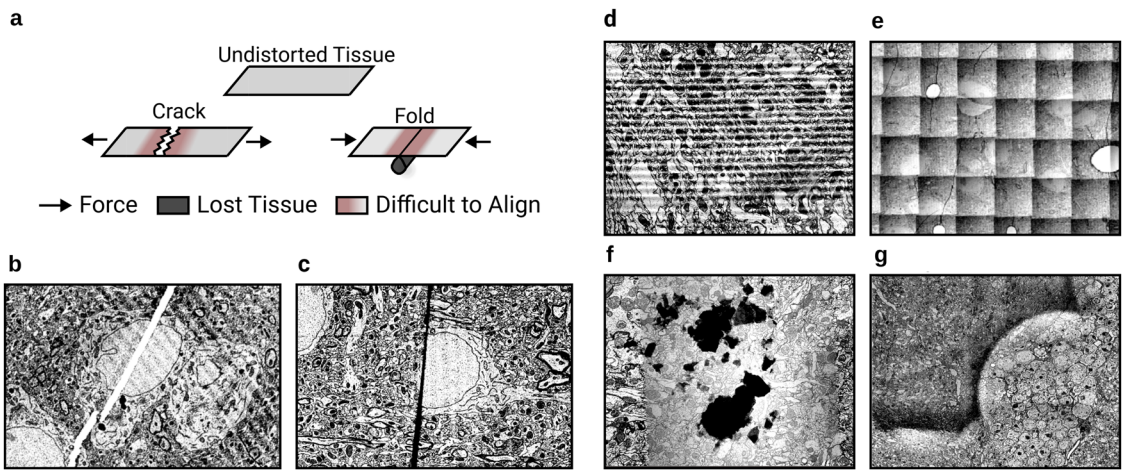

Fig. 1 | Challenges in ssEM alignment. a Mechanics of discontinuous defect creation. Discontinuous defects cause tissue loss and make neighboring areas particularly challenging to align. **b** Example of a crack (white line). **c** Example of a fold (black line). **d** Examples of other artifacts: knife chatter (top left), grid pattern (top right), dust (bottom left), brightness variation (bottom right).

https://doi.org/10.1038/s41467-023-44354-0

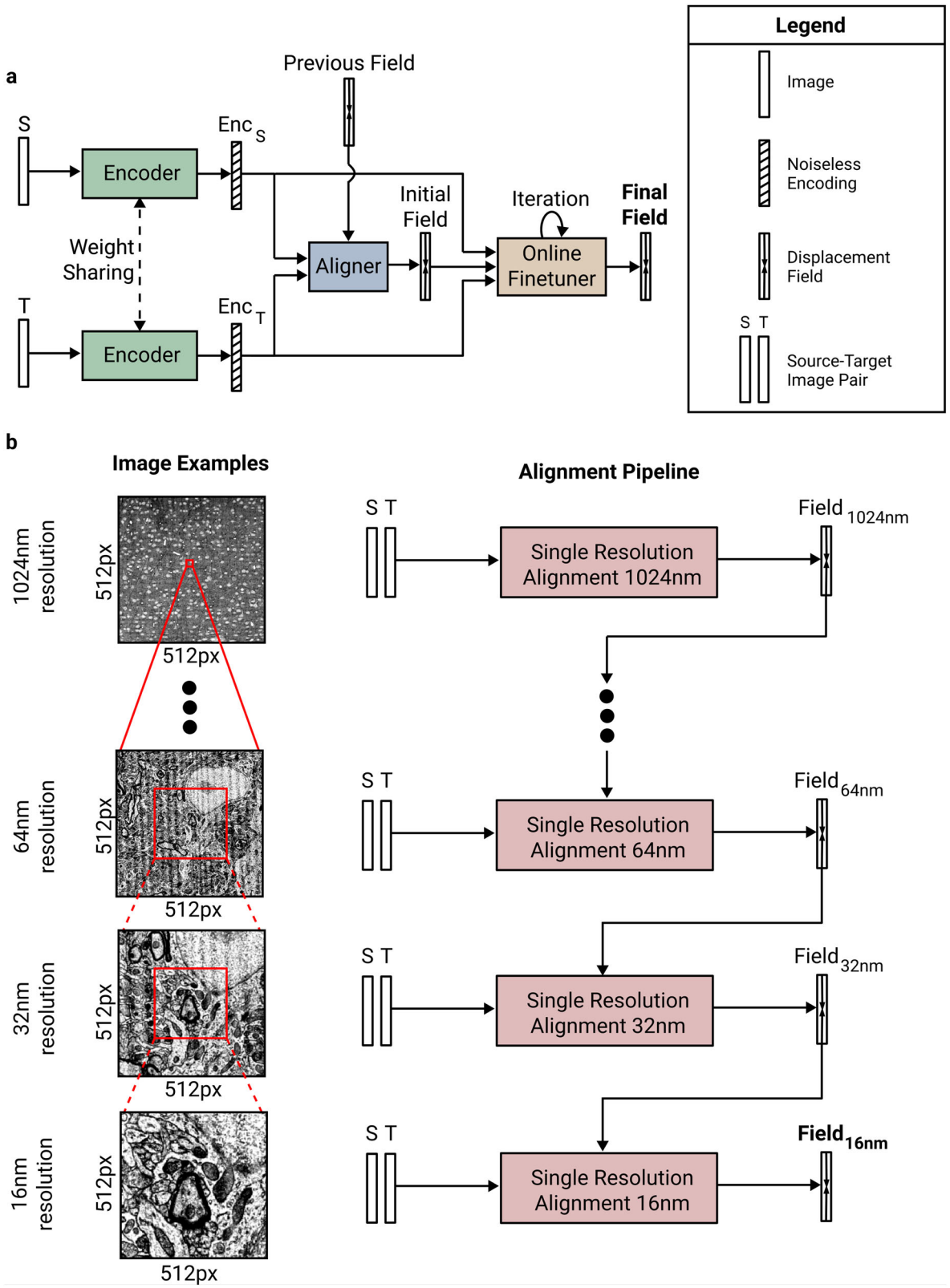

Fig. 2 | SEAMLeSS image pair alignment. a Coarse-to-fine multi resolution alignment. Input to each subsequent stage increases physical resolution, while keeping the same pixel resolution. **b** Architecture of a single resolution alignment pipeline: Encoder, Aligner and Online Field Finetuner.

sections later in the series. This was previously solved by a global relaxation of a spring mesh that extends across the entire series of sections[9]. Here we avoid any global relaxation, which would be computationally expensive for petascale datasets with dense high-resolution transformations. Instead, we compose transformations, but force each transformation to decay in time/depth and in spatial frequency. We call this approach Alignment of Blocks and Composition with Decay (ABCD). ABCD is a computationally efficient and scalable approach, but it can distort morphologies more than global optimization when initial coarse alignment is poor.

https://doi.org/10.1038/s41467-023-44354-0

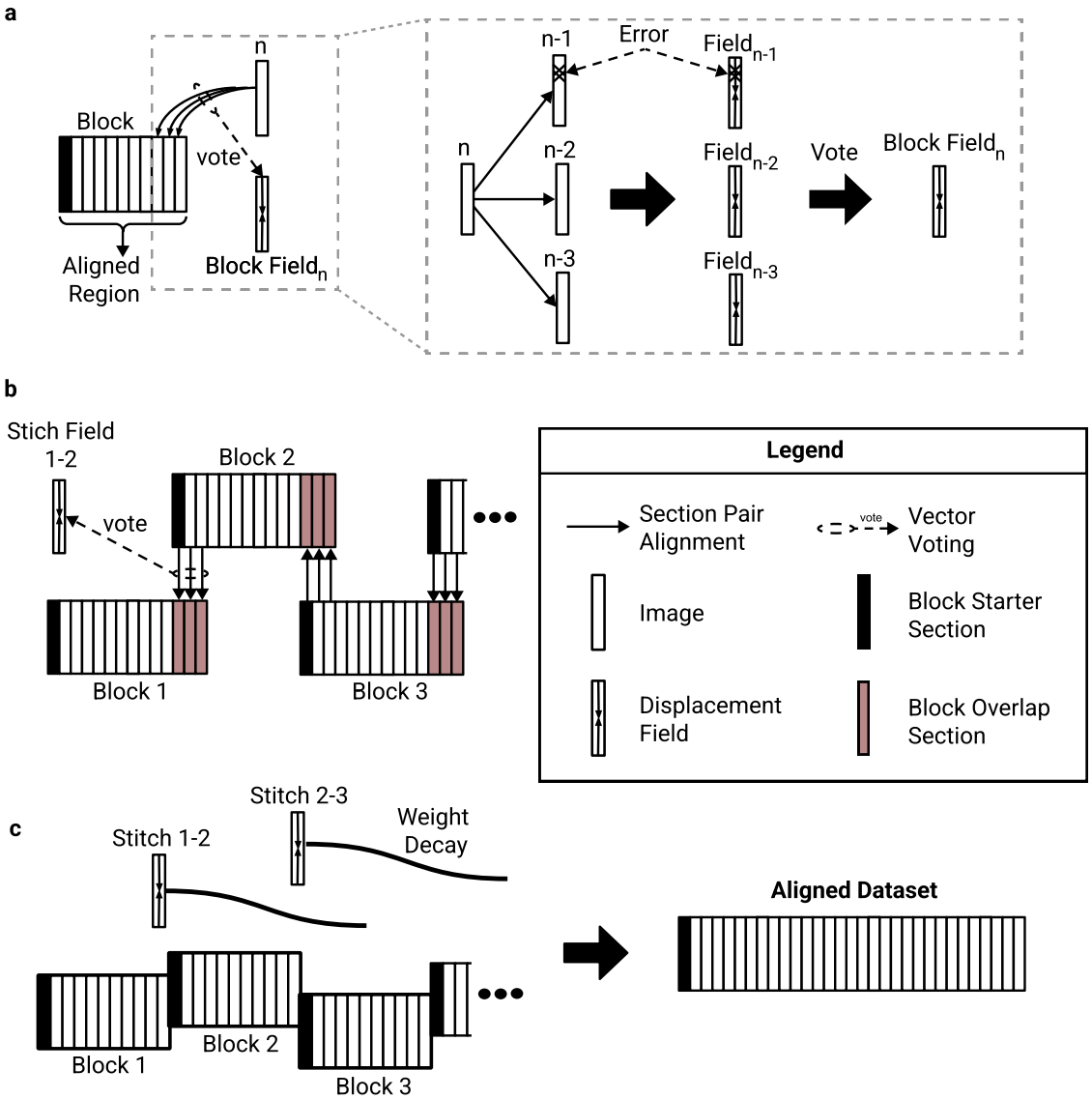

Fig. 3 | Vector voting, distributed block alignment, and composition with decay. a Sequential alignment with vector voting. During block alignment sections are aligned sequentially to the already aligned portion of the stack. Vector voting is performed by aligning the source image to multiple target images, and passing the resulting fields through a pixel-wise smooth median computation. **b** Producing block stitching fields. To produce global alignment, the dataset is broken into overlapping blocks, which are aligned independently. The overlapping regions are aligned to each other to produce a stitching field between each two neighboring blocks. **c** Block stitching. The blocks are stitched together by applying each stitching to successive sections, while reducing its influence with a distance based weight decay.

We release two open-source packages, metroEM and corgie, that together implement our computational pipeline. metroEM implements SEAMLeSS training and inference for image pair alignment. All the multi-resolution Encoder/Aligner pairs (Fig. 2a) can be trained with a single metroEM command. corgie implements large scale global alignment, including block alignment, stitching, and vector voting. corgie is able to handle images that do not fit into single machine memory, provides intuitive tools for distributing tasks to cloud and cluster workers, and can use both metroEM-based or any other user supplied method for image pair alignment. Both packages are implemented using the python (3.6+) programming language and can be installed with the pip package-management system. The released tools enable training of the convolutional nets in our pipeline and application of the convolutional nets to align petascale datasets using cloud Kubernetes or SLURM clusters.

Results

SEAMLeSS image pair alignment

Given a source image and a target image, the task of image pair alignment is to produce a transformation that aligns the source to the target. Transformations are represented as displacement fields, which assign a 2D displacement vector to each pixel (formal definition in Methods). SEAMLeSS works at multiple resolution scales in a coarse-to-fine manner[11,24,25] (Fig. 2a). At each resolution, there is an Encoder net, Aligner net, and Online Field Finetuner (Fig. 2b). The Encoder is applied to both source and target images[26], and the resulting encodings are the inputs to the Aligner. The Aligner computes a residual displacement field, which is composed with the estimate of the displacement field from a coarser resolution, to align the source to the target. The Field Finetuner refines the initial field produced by Aligner through gradient descent to minimize the difference between the image encodings after alignment while keeping the transformation

smooth in the areas without defects[27]. A separate instance of Encoder/Aligner pair is trained for each resolution level.

SEAMLeSS training

The training dataset for Aligner and Encoder nets consists of neighboring image pairs from an unaligned EM stack. Both Aligner and Encoder are trained in a self-supervised manner, meaning that no fine alignment ground truth is required for training. Training is performed sequentially for resolutions in a coarse-to-fine manner.

Training proceeds in two stages. The first stage begins with randomly initialized weights for both Encoder and Aligner. During the first stage, the training loss for both encoder and aligner is based on the squared difference between the images after alignment by the Aligner net, plus an elastic regularizer. (See Methods for precise description of the loss function with Supplementary Fig. 1.) The elastic regularizer penalizes the network from canceling out natural independent motion of biological objects. Since nonsmooth transformations are required at cracks and folds, these defects are detected by another convolutional net, and the regularizer is zeroed out at these locations. The first training stage continues until the loss plateaus As Encoder outputs are not considered during the first stage loss calculation, the Encodings will not suppress artifacts at the end of the first stage. The Encoder/Aligner pair resulting from the first stage will be able to correct most deformations, but generally make errors around image artifacts and other challenging locations.

In the second stage of training, the training loss is based on the squared difference between the encodings (not the images) after alignment by the Aligner net. This can be viewed as metric learning, if the Aligner net is regarded as part of the (trainable) distance function. In other words, the distance function is the Euclidean distance between the encodings (plus an elastic regularizer) after alignment by the Aligner net. Then the goal of the Encoder is to produce encodings that are similar for images that are aligned, and dissimilar for images that are not aligned. The goal of the optimization is to produce image encodings and alignment fields such that the pixel-wise difference between encodings is high before alignment, but low after alignment. Under such training, the Encoder will learn to suppress image artifacts while preserving information about the biological objects. Image artifacts must be suppressed to achieve low pixel-wise difference between aligned encodings because image variation due to artifacts is not generally correlated across the same pixel positions in neighboring aligned sections. At the same time, highlighting biological objects will increase the difference between unaligned encodings. In order to satisfy both training goals, the Encoder ends up suppressing image artifacts and highlighting biological objects.

It is desirable for the encodings to include information about each pixel in the image, as opposed to sparsely representing selected features. With dense encodings, it is possible to precisely correct discontinuous defects based on encoding similarity. We introduce additional training constraints to encourage encodings to be dense as described in Methods. At the end of the second training stage, the Encoder is able to suppress artifacts in EM images.

After a given resolution scale Aligner/Encoder pair has finished training, it is used to process the whole training dataset to produce displacement fields that will be used for subsequent resolution scales training. Training of multi-resolution hierarchies of Aligner/Encoder pairs is implemented in the metroEM package released with this work.

Online field finetuner

Online Field Finetuner refines the initial displacement field through gradient descent. Optimization loss is based on the squared difference between the encodings after alignment plus an elastic regularizer. Similarly to the training stages, the elastic regularizer ensures that natural motion of biological objects between the images will be preserved.

Finetuning by gradient descent is facilitated by using the encodings. Using the raw images instead may cause catastrophic errors because image artifacts are often high contrast, and may end up being the features that are aligned (Fig. 1d). For example, alignment can be corrupted by the parallel stripes of knife chatter (Fig. 1d, top left).

Vector voting

In principle, one could align an entire series of sections by applying SEAMLeSS to align each new section to the previously aligned section. This sequential alignment procedure is well-known[28] and effective if each alignment step is perfect, but lacks robustness to occasional misalignments. These may happen near image artifacts, and are unavoidable in regions where the image data is entirely missing. To increase robustness, we instead align the source image to several last sections of the already aligned portion (Fig. 3a) and apply a pixel-wise smooth median computation over the resulting displacement fields. This Vector Voting operation discards the outlier values for each pixel location of the field, ensuring that only errors that occur at the same pixel location in several neighboring sections will propagate through the block (Methods). Aligning source sections to n target sections is referred to as n-way Vector Voting, where n is referred to as voting distance. For n-way Vector Voting with odd n, only the errors that occur in $(n + 1) / 2$ out of n consecutive sections will be propagated through the block. Increasing voting distance improves error resilience at the cost of increasing the computational cost of alignment.

In alignment methods based on relaxation of a spring mesh[9], springs are sometimes added between next nearest and further neighbor sections, in addition to the springs between nearest neighbor sections. This is a kind of voting, and tends to reduce the magnitude of a misalignment due to a false correspondence but also spreads the misalignment over a larger area. The robustness delivered by Vector Voting should be superior because the effects of outliers are suppressed almost completely.

Distributed alignment of blocks

In order to speed up the computation, we partition the entire series along the cutting dimension into overlapping blocks of contiguous sections. Each block is sequentially aligned with Vector Voting, and the blocks are distributed over independent computational workers.

Vector Voting is not possible for the first $(n - 1) / 2$ sections in a series, which increases the likelihood of error across that region. Therefore adjacent blocks are chosen to overlap by $(n - 1) / 2$ sections, and the first $(n - 1) / 2$ sections of each block are discarded before combining the blocks to create a global alignment (Fig. 3b), as will be discussed next.

Composition with decay

After all the independent blocks are aligned, they should be stitched together into a single globally aligned stack. We define the stitch field between block N and block $N + 1$ as that obtained by aligning the overlapping section region of block $N + 1$ to the same sections in block N and performing vector voting operations across the obtained fields (Fig. 3b). The stitch fields can be used to create a global alignment as follows. For each block, the stitch fields from the preceding blocks are composed to create an accumulated transformation that is applied to all sections in the block (Fig. 3). This straightforward procedure can result in good alignment, in the sense that neurites follow smooth paths through the stack. However, because each stitch field represents a non-rigid transformation, the composition of many stitch fields can accumulate errors and result in a high frequency field that distorts cell morphology (Fig. 4h).

Therefore we modify the procedure as follows. When applying a stitch field to subsequent sections, we decrease its weight relative to the distance from the block interface (Fig. 3, Methods). The number of sections that each stitch field influences is referred to as *decay*

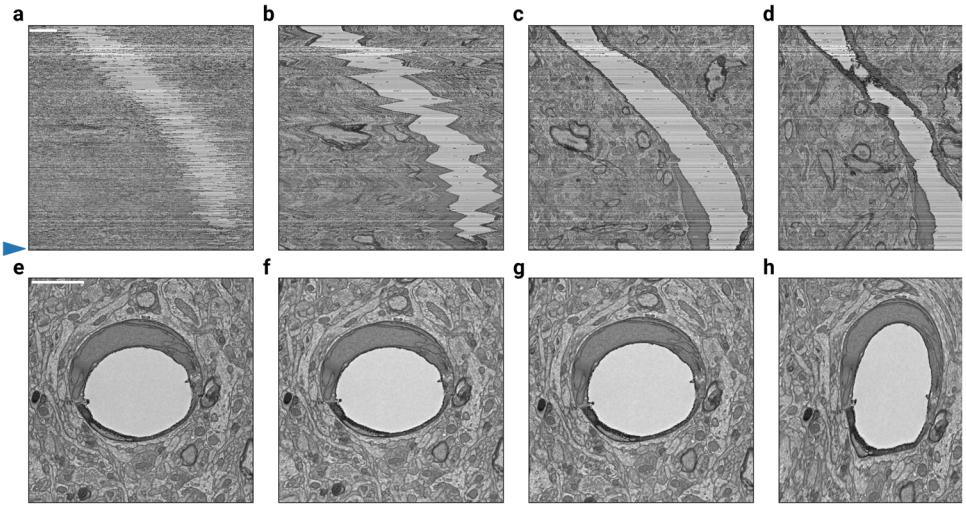

Fig. 4 | Alignment of blocks and composition with decay. A stack of 450 mouse cortex sections[5] was block aligned (10 sections per block), then globally aligned by composing the linearly decayed stitching fields between blocks with varying decay distances. **a**–**d** The rough aligned sections, the aligned sections with decay distances of 10, 100, and 500 sections. The vertical axis is orthogonal to the cutting plane, while the horizontal axis is in the plane. **e**–**h** The 447th section of the sample above (see blue arrow along sides of (**a**–**d**)), approximately centered on the same blood vessel to demonstrate the distortion that results from composing many fields together without decay. The horizontal axes of the top and bottom row are the same. Scale bars are 2 μm.

distance. Practically, decaying distances of 100-300 sections avoid excessive drifts (Fig. 4b,f) while also preventing morphology distortions (Fig. 4h). Figure 3c and g show an example alignment with decay distance of 100.

Alignment of Blocks and Composition with Decay (ABCD) avoids a global optimization with an offline solver, which becomes computationally expensive with saturated high-resolution fields[8,9].

Whole fly brain alignment

We evaluate our alignment pipeline by applying it to the full adult fly brain (FAFB) dataset[3]. As a baseline, we first consider the original alignment (v14), which was created using a piecewise affine approach[8]. We evaluated alignment accuracy on a cutout of 1000 full sections using the following procedure. First, we divide neighboring image pairs into 64 x 64 px chunks at 32 nm pixel resolution and compute Pearson Correlation for each chunk, a block-based quality assessment that we refer to as the Chunked Pearson Correlation or CPC[29]. High CPC values indicate well-aligned sections, while low CPC values indicate differing image contents between neighboring sections, which could signify a potential misalignment, as well as natural motion of the objects or a visual artifact (Fig. 4d).

To determine misaligned locations, we sampled 300 locations with CPC <0.25, and manually classified the cause of low CPC for each location (see Methods). We estimate that 78% of locations with CPC <0.25 in v14 are caused by genuine misalignments. Extrapolating to the total number of low CPC locations, 3.31% of total tissue area in v14 are misaligned.

While the great majority of v14 is well-aligned, the misaligned portion poses challenges for downstream image processing tasks[30]. Therefore we were motivated to create a v15 alignment of FAFB using the pipeline presented in this paper. The alignment was performed from sections that were coarsely registered to v14 with a near-rigid alignment at 512 nm resolution. Only 0.51% of v15 has CPC <0.25 (Fig. 5a, b), and we estimate that just 0.06% of v15 contains a genuine misalignment. This is almost two orders of magnitude better than v14.

Not all of the v15 improvement can be attributed to alignment, because the 2D section images for v15 were reworked to remove montaging errors in v14. However, only a small fraction of total area (<1%) was directly affected by montaging errors. Moreover, most montaging errors occurred around tissue boundary, while our evaluation shows that v15 specifically outperforms v14 at correction of discontinuous defects inside the tissue (Fig. 5b, Supplementary Fig. 2). As an additional effort to isolate the improvements due to alignment rather than montaging, we evaluated another FAFB alignment (v14.1) that was previously released[31]. v14.1 was created by applying a preliminary version of our pipeline (Methods) to v14 images, so that montaging errors in v14 were "baked in" and could not be corrected. We estimate that 0.58% of v14.1 contains a genuine misalignment with CPC <0.25. This is almost 6 × better than v14, suggesting that some of the improvement in v15 is due to alignment alone. At the same time, v14.1 alignment performs significantly worse around discontinuous defects (Fig. 5b), which highlights the importance of pipeline components introduced in this work.

Petascale alignment of mouse cortex

A preliminary version of our pipeline, similar to the one used for the v14.1 FAFB alignment, was applied to two mouse cortex volumes amounting to over 1.1 mm^3. The largest mouse cortex dataset[5] was composed of 19,939 square millimeter sections imaged at 4 nm resolution (Supplementary Fig. 3). Numerous cracks and folds occurred in every section, posing challenges more severe than in FAFB (Supplementary Fig. 4). The discontinuous distortion introduced by cracks and folds ranged up to roughly 10 μm, with the cross-sections of neurites nearly 1000 × smaller than the distortion. Even our preliminary pipeline was able to precisely correct such large discontinuous distortions, ensuring that neurites at the boundary of the defect were still well-aligned (Supplementary Figs. 4, 5). This led to a successful automated reconstruction[32]. The alignment took 230 h on a cluster of preemptible NVIDIA T4 GPUs on Google Cloud. The cluster size fluctuated due to hardware availability constraints, with the number of active GPUs averaging at 1200.

Discussion

Our pipeline computes saturated displacement fields, rather than the sparse correspondences of traditional approaches[8,9]. This enables the representation of nonsmooth transformations, which are necessary for accurate alignment near cracks and folds. Displacement fields are initially predicted by the SEAMLeSS convolutional nets, and then improved by the Online Field Finetuner. This combination is more reliable at avoiding false correspondences than traditional approaches

https://doi.org/10.1038/s41467-023-44354-0

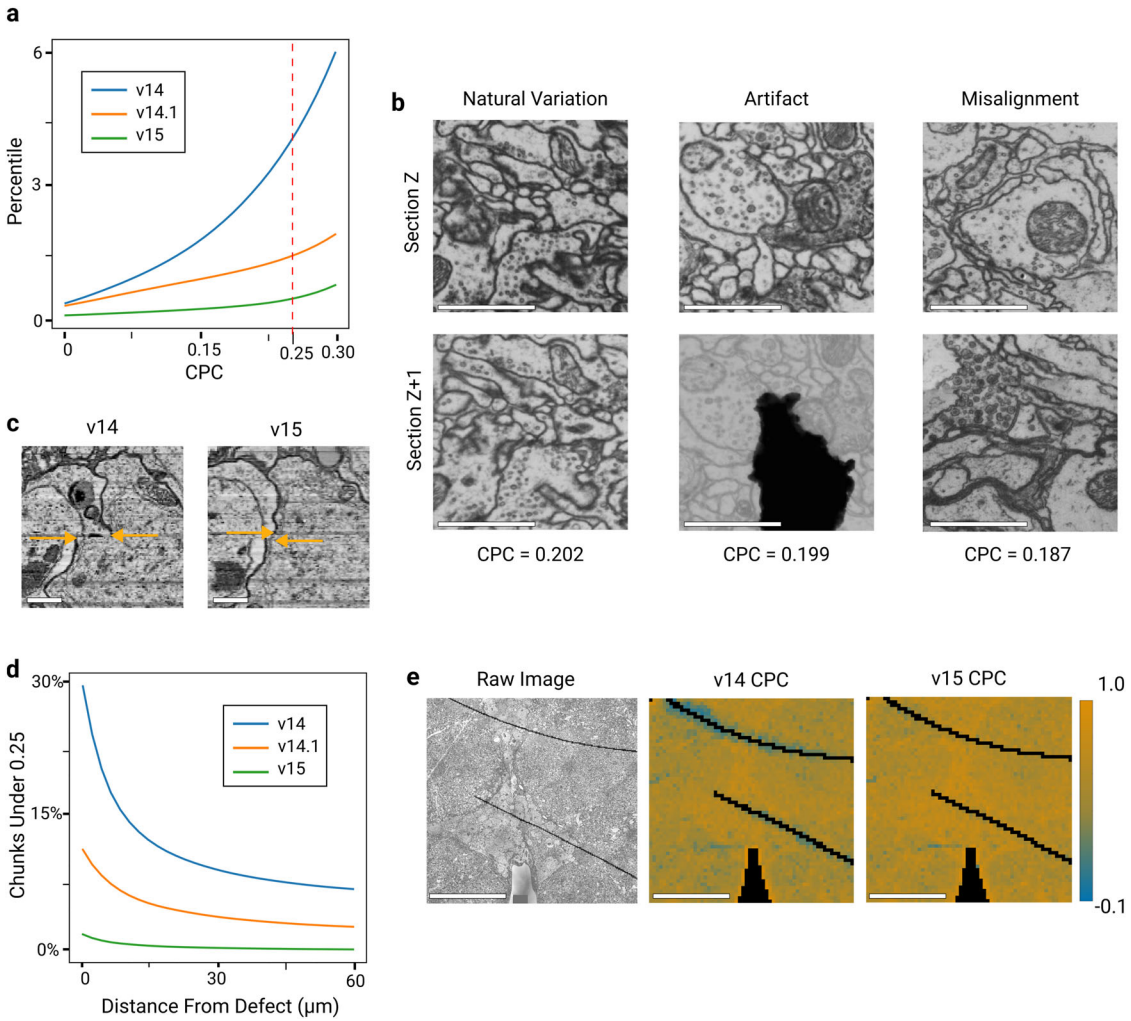

Fig. 5 | FAFB alignment quality. The v15 alignment of the FAFB dataset improves over v14.1[31], which in turn improves over the original v14[3]. Accuracy is quantified using the Chunked Pearson Correlation (CPC), which tends to be high at well-aligned locations. The evaluation was conducted for a range of 1000 sections. (**a**) CPC percentile decreases with progressive improvement of the three alignment versions. Chunks with low CPC (<0.25, dashed red line) are just 0.51% of v15. CPC was computed for $(2048\ nm)^2$ chunks at 32 nm pixel resolution, and non-tissue chunks and chunks that include a discontinuous artifact were excluded. **b** Estimated misalignment rate based on manual annotation of low CPC chunks. v15 improves over v14 by two orders of magnitude (rightmost column). CPC may be low for reasons other than misalignment, so 300 locations with CPC < 0.25 were randomly sampled from each alignment version, misalignments were identified by a human expert, and the results were extrapolated to the entire image stack. Note that v15 contains fewer low CPC locations caused by "Natural Variation" because it has less consistent drift compared to v14 and v14.1 (**c**) Example misalignment in v14 corrected in v15. Arrows indicate cell body boundary position, and the horizontal space between the arrow heads indicates misalignment. Horizontal and vertical axes are parallel and perpendicular to the sections, respectively. Scale bar 1 μm. **d** Examples of low CPC (<0.25) chunks caused by natural variation of image content, image artifact and misalignment, the categories that were manually annotated in (**b**). Note that a misalignment of a large magnitude was chosen for visual clarity. Scale bar 1 μm. **e** Percentage of low CPC (<0.25) chunks as a function of distance from a discontinuous defect. In v15, such defects cause little increase in low CPC chunks. **f** Single section CPC heatmap comparison between v14 and v15. Black corresponds to non-tissue and discontinuous defect chunks, which are ignored in (**a**). Scale bar 50 μm.

based on Block Matching or SIFT features, as shown by our evaluation on the FAFB dataset.

The rate of false correspondences can be reduced in traditional approaches by various postprocessing schemes[9], but even a low rate amounts to a large absolute number of false correspondences in a sufficiently large dataset, as in the FAFB v14 alignment. For a previous terascale alignment, we eliminated the remaining false correspondences by manual intervention[15], but the required labor would be prohibitive at the petascale. Our approach is to drastically reduce the rate of false correspondences by training the SEAMLeSS nets.

Preliminary version of the presented pipeline used to produce FAFB v14.1 alignment Dorkenwald, et al.[31] used a network architecture presented in Hui, et al.[33] in combination with loss drop similar to Yoo, et al.[17], but applied specifically to defects detected by a convolutional net[23]. As described in Mitchell, et al.[16], the preliminary version of the pipeline struggled with large displacements, visual artifacts, and catastrophic errors. When creating v15, we found that the Finetuner is helpful for improving the accuracy of alignment, especially near folds and cracks. Also, the Finetuner reduces the accuracy requirements for the SEAMLeSS nets, making training them easier.

Conversely, one can imagine eliminating SEAMLeSS from the pipeline, and using the Online Field Finetuner only, but this would lower accuracy for two reasons. By providing a good initial guess for the displacement field, the SEAMLeSS nets prevent the Finetuner from getting trapped in a bad local optimum[16]. Furthermore, the SEAMLeSS Encoder suppresses image artifacts (Fig. 6), which otherwise can cause misalignments to become global optima.

Our ABCD procedure provides a computationally efficient alternative to traditional approaches involving global optimizations over all sections in an image stack[9]. All ABCD computations are local, involving

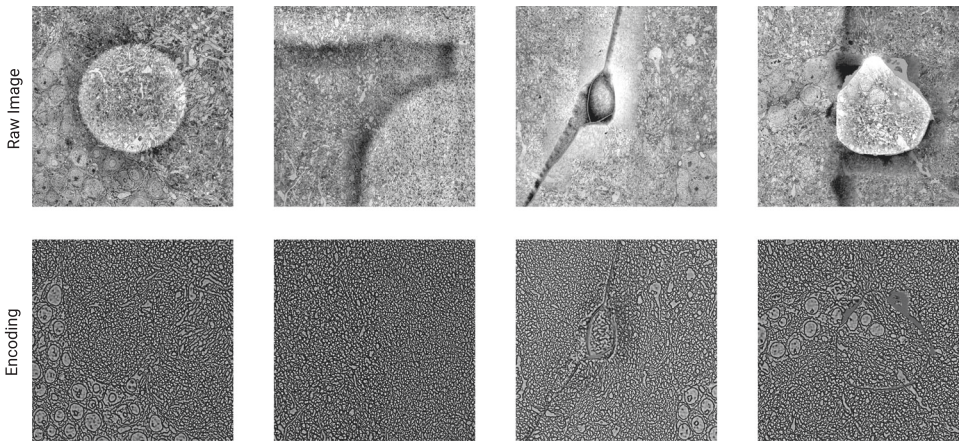

Fig. 6 | Encodings suppress artifacts. SEAMLeSS is able to produce dense Encodings that suppress many artifacts present in ssEM images. As a result, Encodings are more suitable than images as inputs to the Online Field Finetuner.

neighboring sections only. Within each block of sections, a section is aligned to the previous n sections considered by the Vector Voting procedure. When stitching blocks together, blocks do not interact with other blocks beyond the decay distance. Such locality enables graceful extension to petascale datasets, and the blocks can be processed in parallel by hundreds of distributed workers. At the same time, ABCD has several shortcomings. ABCD introduces directional bias and makes the alignment outcome dependent on the starter sections. We practically find that suitable starter sections can be effectively identified by manual annotation, although this process can be automated. ABCD relies on the elastic regularizer used during image pair alignment to conserve natural motion of biological structures.

The maximum deformation magnitude that SEAMLeSS can correct depends on the resolution of the coarsest Encoder/Aligner pair in the pyramid. In the presented experiments, the maximum Encoder resolution was set to 8 µm, which led to consistent correction of discontinuous defects up to 10 µm and near-rigid deformations up to 20 µm. Such displacements could only be corrected in tissue islands that constituted >100 pixels at 8 µm resolution. Inability to align tissue islands whose size is small relative to the magnitude of the displacement is an inherent limitation of SEAMLeSS, as it becomes too difficult for the nets to produce informative encodings when the whole island is only a few pixels wide. For such cases, SEAMLeSS has to rely on conventional methods that perform initial coarse alignment.

The use of deep learning based dense features for ssEM alignment was first proposed by Yoo, et al.[17]. Autoencoders attempt to preserve all of the information in the image including visual artifacts, unless specifically designed training data augmentations are used. In contrast, our encodings are trained with metric learning and learn to suppress visual artifacts without any supervision.

In the preceding, we have emphasized the replacement of traditional computer vision by our approach. In reality, the approaches are complementary and can be combined. This is already assumed by the present paper, as our pipeline is applied only after the images have been coarsely aligned using traditional computer vision approaches (Methods). Although our approach does not require a global spring mesh extending over all sections, an elastic regularizer is used for 2D patches during SEAMLeSS training and by the Online Field Finetuner at runtime. One can easily imagine other hybrid methods. For example, Block Matching should be more robust to image artifacts if applied to our SEAMLeSS encodings rather than to raw images or linearly filtered images. Our ABCD procedure, including Vector Voting, could be applied to transformations computed by Block Matching rather than SEAMLeSS. These examples of hybrid methods are hypothetical, and are left for future work.

Methods

Summary of FAFB v14.1 and v15 pipeline differences

The v15 image pair alignment method described in this paper has the following major difference with the v14.1 method described in our preprint[16]. First, v14.1 does not use the Online Finetuner, and so is less robust than the v15 version. The training loss used in v14.1 uses raw image based similarity, while v15 loss uses raw image based similarity for the first stage and the encodings for the second stage. Consequently, v15 loss is less affected by visual artifacts, providing a cleaner training signal. Because v14.1 uses the network architecture proposed by Hui, et al., 2018, the v14.1 does have an encoding component. However, the no constraints are put on v14.1 encodings during training, and so the encodings are not incentivized to suppress visual artifacts or to produce dense features. Consequently, the v14.1 encodings are sparse and do not suppress visual artifacts, and so cannot be effectively used to guide Online Finetuner. Additionally, v14.1 performs alignment with a single network where encodings at lower resolutions depend on the encodings at higher resolutions. With the v14.1 approach, lower resolution modules cannot be trained separately from high resolution modules, which limits training time patch size and inhibits the network's ability to correct large displacements. v15 breaks this architecture into independent components at different resolution levels. v15 approach has been used to correct displacements of up to 20 microns, while the v14.1 approach could only correct displacements of up to 1 micron.

v14.1 and v15 used identical alignment globalization approaches – both used ABCD with vector voting. The v14.1 pipeline used Vector Voting with a voting distance that was manually set for each section. The voting distance was typically 5, with some small regions set to 7 or 9, allowing additional computation to be spent for specific sections that needed to contend with long stretches of missing data. v15 pipeline used smaller vector voting distance of 3 for two reasons. First, v15 produced a much lower rate of misalignments, and so fewer errors needed to be fixed. Second, v15 did not consider empty (black) regions as valid voting targets and would always perform voting across nearest 3 valid sections. This helped v15 effectively handle large stretches of missing data without increasing the voting distance.

FAFB v14.1 vs v15 montages

Every section image is composed of multiple image tiles, which are combined by a process known as montaging, stitching, or mosaicking. Montages in v14 were generated using SIFT feature matches, geometric consensus filtering and global optimization of per-tile affine transformations[3,8,9]. Many montages in v14 have substantial stitching errors. Many v14.1 problems were the result of errors in v14 that could

not be undone by our alignment pipeline. To eliminate these errors, we attempted to produce a clean set of montages that would serve as the starting point for our alignment pipeline.

We enhanced the EM_Aligner montage diagnostic tools (EM Aligner montage diagnostic tool: https://bit.ly/3gI5Cg0) and found that 2,871 out of 7,050 montages had substantial point-match residuals (≥1% of all tile pairs having an error ≥ 40 nm, the cellular membrane thickness ≈ 20 nm). Of the remaining 4,179 montages 4,074 consisted of disconnected 'islands' that were improperly interleaving which corrupted the montages.

We improved the quality of point matches for stitching by re-parameterizing both feature extraction and geometric consensus filters. We used five separate sets of parameters (Montage SIFT parameter sets: https://bit.ly/3sy44Lf) and iterated through them sequentially until a sufficient number of matches were derived. This iterative approach allowed us to generate most matches quickly and spend more compute time only where necessary. Remaining unmatched tile pairs were manually reviewed and we tried different custom match derivation parameter sets to connect them. Often, the most difficult to match pairs contained large areas of featureless resin in their overlapping region.

We further improved the global optimization of per-tile transformations. We used the EM_aligner solver[8] that was used for the v14 reconstruction and later the Python implementation BigFeta (BigFEaTure Aligner (BigFeta): https://github.com/AllenInstitute/BigFeta). For montages where these solvers did not produce satisfying results, we used the iterative TrakEM2 solver[9] (TrakEM2 solver client: https://bit.ly/3gCuu98) that enables rigid regularization of per-tile affine transformations but is significantly slower. This was primarily necessary for the 295 sections that contained images from multiple acquisitions.

We separated the disconnected islands in all 7,050 sections resulting in 48,121 sub-sections. These sub-sections were then rigidly registered to the corresponding sub-section in the FAFB v14 aligned data set[3]. Islands that were not present in the v14 data set, were placed based on stage coordinates of the microscope. This provided a good starting location for the cross section alignment process, essentially using v14 results to roughly align the entire volume.

As in v14, we used the Distributed Multi-Grid library[14] to intensity correct all data and corrected a number of previously undetected mistakes.

With the updated montaging pipeline, the FAFB montage series consisting of 14,442,143 tiles can be generated in ≈ 200,000 CPU hours.

FAFB v15 coarse alignment

Before using ABCD, we had to perform a round of coarse alignment for two reasons. First, the rigidly registered sections contained a large number of section-wide scale changes. As scale changes are non-rigid transformations, they would be penalized by the elastic regularizer used in SEAMLeSS and lead to poor image pair alignment quality. Moreover, because many of FAFB islands were overlapping after the rigid registration, each section was represented with several images, one per island, to avoid data loss. This posed an operational challenge for our implementation of ABCD, as it only supported aligning stacks of single images, not sets of potentially overlapping images. It was practically the easiest solution to first remove overlaps by coarsely aligning each of the islands to the corresponding v14 section, then combine each islands for each section into a single mage, and then proceed with fine alignment using our implementation of ABCD. Note that this is the only way in which v15 used v14 alignment – as a reference for the initial coarse alignment.

Two different alignment approaches were used for different section ranges. Sections in ranges 1–1500 and 5850–7060 frequently contained scale changes, inspection showed that they didn't have non-affine deformations >20 um. All of the islands in this range of sections were aligned to the corresponding v14 section using affine approach. The middle range of sections Z 1500–5850 didn't contain scale changes, but contained some large scale non-affine deformations. One of the failure modes was for one portion of the section to rotate while another portion counter-rotated. Attempting to correct such deformations with conventional template matching approaches required a large amount of manual intervention. Instead, we used SEAMLeSS with 8192 nm and 512 nm encodings as our main method for island alignment. Encodings were trained on 1000 consecutive sections form Z range 2000–3000, and generalized well to the whole dataset. Aligner nets were not used for coarse alignment as applying Online Finetuner to such coarse resolutions was computationally cheap, and also the 8192 nm aligner tended to overfit to the training dataset. This method was able to correct all of the non-affine deformations in the main island, but produced bad alignment for small islands that couldn't be adequately represented at 8192 nm resolution. In order to handle both cases, two independent alignment versions were produced for each island – one using the Online Finetunner approach and the other using affine alignment. The alignment version that produced higher similarity to v14 was chosen for the final alignment. Similarity was calculated by applying Multiscale-SSIM[34].

Modifying the elastic regularizer used during SEAMLeSS image pair alignment would allow us to avoid using affine approach for sections in 1–1500 and 5850–7060 ranges. This development is left for future work. Inability to align tissue islands whose size is small (<10x) relative to the magnitude of the displacement is an inherent limitation of SEAMLeSS. It becomes too difficult for the nets to produce informative encodings at coarse resolutions that make the whole island only a few pixels wide. For such cases, SEAMLeSS has to rely on conventional methods that perform initial coarse alignment.

FAFB v15 fine alignment

After coarse alignment, combined islands for each section into a single image and corrected scale changes and deformations beyond 20 um, we fine aligned it using ABCD, vector voting and Encoder/Aligner pair described in this paper. Image pair alignment was performed using 4 Aligner/Encoder pairs at 512 nm, 128 nm, 64 nm and 32 nm. The training set for 512 nm was composed out of 1000 consecutive sections taken from Z 2000–3000. The training set for the 128 nm, 64 nm and 32 nm were manually selected. A trained operator selected 435 locations within the image that contained visual artifacts patterns. At each of these locations, cutouts of the same spatial region were made for image pairs between z & z-1, z-4 & z-5, z-9 & z-10, & z-14 & z-15, so that there were 1740 image pairs included in the training set. Each Aligner/Encoder pair was trained for 600 epochs (100 epochs first stage + 500 epochs second stage), which took 72 h on a single Ampere class NVIDIA GPU.

Alignment globalization was performed using ABCD with blocks of 25 sections. Starter sections were selected manually using the following process. A human annotator inspected every 25th section, flagging ones that had large discontinuous defects or other significant corruption. It took about 1 h to inspect 282 sections, 15% of which were determined to be unsuitable as a starter section. The neighborhood of each flagged section was inspected to find a substitute for the block start. The final set of suitable starter sections was independently reviewed by another inspector to avoid potential human error. We opted to perform this process manually rather than automate it because it only took around 6 human hours to complete. There were 3 sections of overlap between blocks that were used to produce the stitch fields. A decay distance of 100 sections was used to compose the blocks into a globally consistent alignment. Within each block, 3-way vector voting was used. Parameters were selected that did not distort the tissue on a large-scale (Supplementary Fig. 6). Missing data regions were not considered to be a valid vector voting target, so voting was performed only on regions that contained tissue. Alignment for the

whole FAFB (7052 sections) performed on a cluster with 128 Ampere class NVIDIA GPUs and and the final displacement fields were produced in 240 h.

FAFB misalignment proofreading rubric

Classification of locations with CPC under 0.25 was performed by expert human annotators. The cause of low correlation at each location was assigned one of three classes – Visual Artifact, Natural Variation, and Misalignment. For each alignment version, we randomly sampled 300 locations out of the set of all chunks with CPC under 0.25. Classification was performed according to the following criteria:

1. Location where one of the two neighboring images included artifacts or tissue corruption that prevents clear tracing of biological objects are classified as Visual Artifact.
2. Locations with relative motion sufficient to break biological objects, which is consistent with motion of the same objects in preceding and following sections, and where the motion corresponds to small areas or naturally fast moving structures, such as axon bundles and tissue boundary, are classified as Natural Variation.
3. Locations with relative motion sufficient to break biological objects which is not consistent with motion of the same objects in the preceding and following sections are classified as Misalignment.
4. Locations with relative motion sufficient to break of biological objects which is consistentf which with motion of the same objects in preceding and following sections, and where the motion corresponds to a large area of not naturally fast moving structures are classified as Misalignments.
5. Locations that do not qualify for criteria 1–4 and contain visual artifacts such as contrast and brightness variation are classified as Visual Artifact.
6. Locations that do not qualify for criteria 1–5 are classified as Natural Variation.

Proofreading annotations for each location along with the CPC maps and random seeds used during sampling are included with this submission.

Petascale mouse cortex

Serial section EM images of a cubic millimeter volume of mouse cortex were acquired at the Allen Institute[4,5]. The images were aligned using a preliminary version of our pipeline.

Mouse coarse alignment

The connectomics team at the Allen Institute for Brain Sciences stitched and aligned the image stack to displacements within 20 μm using their ASAP framework[35,36].

The coarse alignment was further refined using an Encoder trained to optimize patch matching correlograms for 1024 nm data as described in Buniatyan, et al., 2020. The Aligner was a blockmatching method that correlated 256 px patches of the source image to 358 px patches of the target image on a Cartesian grid of 96 px. The dataset was divided into blocks of 100 sections, sequentially aligned with 5-way Vector Voting, and stitched using a decay distance of 300.

Mouse fine alignment

This preliminary version of the pipeline did not include an Encoder, but it did make use of the Online Field Finetuner. The blockmatched dataset was further aligned using an Aligner trained on 1024 nm data, and that output was again aligned by an Aligner trained on 64 nm data. This approach of aligning the entire dataset at a lower resolution before proceeding to higher resolutions has been replaced in the current pipeline with multi-resolution Image Pair Alignment.

The input to the 1024 nm Aligner was the output of the Encoder used during coarse alignment. The input to the 64 nm Aligner was raw images, with no Encoder. The 64 nm Aligner was trained on a curriculum of example images containing manually curated image artifacts.

v15 pipeline technical details

The current pipeline was used to produce the v15 alignment of the FAFB dataset, as well as the alignment of a small sample of the petascale mouse cortex dataset that was previously aligned using a preliminary version of this pipeline[5] (Supplementary Fig. 7).

Semantic segmentatio

Samples of manually annotated cracks and folds in cutouts of 1024 x 1024 px at 64 nm resolution, were collected from the original data and were used to supervise training of separate UNets as described in Macrina, et al., 2021. A tissue segmentation model was trained in similar fashion. Segmentations were downsampled with min pooling, to overestimate non-tissue regions at lower resolutions.

Image preprocessing

Prior to alignment training and inference, the ssEM image stack is downsampled, masked, and normalized. First, the original images are downsampled through average pooling to produce a hierarchy of resolutions ranging up to 4096 nm. Next, discontinuous image defect masks and resin masks are produced by specialized convolutional networks. Salient visual features of cracks, folds, and resin make detection a relatively simple task. Masks are typically detected at 96 nm resolution. Note that both false positives and false negatives during defect detection will negatively affect the final alignment quality. However, such mistakes are typically rare as both cracks and folds are marked by salient visual features. After the masks are produced, they're downsampled to lower resolutions through max pooling. At each resolution, ssEM image pixels marked by the masks are zeroed-out, thus removing them from the image. Finally, the images are normalized at each resolution. Zero valued pixels, meaning pixels corresponding to empty space or the masked regions, are ignored during normalization. This means that mean and variance values used during normalization are computed based on values of non-zero pixels only, and zero valued pixels remain zero valued after the normalization. The final normalized stack of multi-resolution images with zeroed-out defect and resin regions is used during all of the alignment steps.

Transforming images with displacement fields

Due to the large anisotropy in serial sectioning data, we consider the data as a series of 2D images, $M : \mathcal{R}^2 \to \mathcal{R}$. We define a displacement field, $\boldsymbol{F} : \mathcal{R}^2 \to \mathcal{R}^2$, in the transformed space, $\vec{r}$, with a displacement, $\boldsymbol{u}(\vec{r})$, that indicates a position in the initial space, $\boldsymbol{F}(\vec{r})$, such that

$$\boldsymbol{F}(\vec{r}) = \vec{r} + \boldsymbol{u}(\vec{r}). \tag{1}$$

An image M is transformed by a displacement field by sampling,

$$(M \circ \boldsymbol{F})(\vec{r}) = M(\vec{r} + \boldsymbol{u}(\vec{r})). \tag{2}$$

The aim of alignment is to find a displacement field that will transform each section to remove distortion.

Displacement fields, $\boldsymbol{A}(\vec{r}) = \vec{r} + \boldsymbol{a}(\vec{r})$ and $\boldsymbol{B}(\vec{r}) = \vec{r} + \boldsymbol{b}(\vec{r})$ can similarly be transformed or "composed", such that

$$(\boldsymbol{A} \circ \boldsymbol{B})(\vec{r}) = \vec{r} + \boldsymbol{b}(\vec{r}) + \boldsymbol{a}(\vec{r} + \boldsymbol{b}(\vec{r})). \tag{3}$$

We used bilinear interpolation kernels to implement composition and sampling discretely[37]. We may leave off the indexing with $\vec{r}$ for readability and write these transformations as $M \circ \boldsymbol{F}$ or $\boldsymbol{A} \circ \boldsymbol{B}$.

Loss masking

As described in Image Preprocessing section, locations in the image that are non-tissue are set to zero. It is desirable to ignore both the similarity and the deformation loss components at such locations in the image. Given an image M, we define a non-zero coordinate set T as

$$T(M) = \{\vec{p} \mid \vec{p} \in \mathbb{R}^2, M(\vec{p}) \neq 0\}. \quad (4)$$

We will refer to $T(M)$ as a set of tissue coordinates of M. This is accurate because the zeroed-out regions of the image do not correspond to tissue measurement anymore. As a shorthand, we will denote $T(M_s \circ \boldsymbol{F}) \cap T(M_t)$ as T_{s+t}.

We also define a masked average operator ψ to average all values of an image K over a given set of locations P,

$$\psi(K,P) = \frac{1}{|P|} \sum_{\vec{p} \in P} K(\vec{p}). \quad (5)$$

Elastic regularizer

For each pixel coordinate $\vec{p}$, we define a set of neighbors $N(\vec{p})$ as

$$N(\vec{p}) = \{\vec{p} + (\vec{a,b}) \mid a,b \in \{0,1\}, (a,b) \neq (0,0)\}. \quad (6)$$

We define elastic energy of a vertex at coordinate $\vec{p}$ as the sum of elastic energies connected to that vertex. The elastic energy map Ω for a field $\boldsymbol{F}$ is a mapping between pixel coordinates and the elastic energy of the vertex associated with that coordinate,

$$\Omega(\boldsymbol{F})(\vec{p}) = \sum_{\vec{h} \in N(\vec{p})} \left(\| \boldsymbol{F}(\vec{p}) - \boldsymbol{F}(\vec{p} + \vec{h}) \| - \| \vec{h} \| \right)^2. \quad (7)$$

We define elastic regularizer L_{er} as the masked average of elastic energies for tissue vertices in the source image,

$$L_{er} = \psi(\Omega(\boldsymbol{F}), T(M_s \circ \boldsymbol{F})). \quad (8)$$

Image pair alignment

For a given resolution, the inputs to an image pair aligner are a source and target image, M_s, M_t, and the output is a displacement field $\boldsymbol{F}$ and image encodings Q_s and Q_t. The aligner is trained in two stages with two loss functions. During the first training stage, the loss is formulated as combination of similarity between images and elastic regularizer. We define a pixel-wise square error map between images M_1 and M_2 as $E(M_1, M_2)$,

$$E(M_1,M_2)(\vec{p}) = (M_1(\vec{p}) - M_2(\vec{p}))^2. \quad (9)$$

The training loss during the first training stage, L_1, is defined as a combination of mean squared error between M_t and $M_s \circ \boldsymbol{F}$ masked over T_{s+t} and the elastic regularizer,

$$L_1 = \psi(E(M_s \circ \boldsymbol{F}, M_t), T_{s+t}) + \gamma L_{er}, \quad (10)$$

where γ is a hyperparameter that balances similarity and deformation loss. In our experiments γ varied from 5 to 25, typically with a higher value for lower resolution aligners. The first stage of training proceeds for 100 epochs.

After the first stage of training, the Encoder/Aligner pair is able to produce an initial displacement field, but the image encodings may still contain visual noise. The second stage of training removes visual noise from the image encodings, increases density of the encoding features, and improves the initial displacement field produced by the Aligner.

In the second training stage, the loss is computed as a combination of three components: post-alignment similarity, pre-alignment similarity, and elastic regularizer. On a high level, post-alignment similarity minimizes the difference between encoding after alignment, thus encouraging producing accurate alignment field $\boldsymbol{F}$ as well as suppressing noise in the encodings.

The post-alignment loss component L_{post} as mean squared error between $Q_s \circ \boldsymbol{F}$ and Q_t masked over T_{s+t},

$$L_{post} = \psi(E(Q_s \circ \boldsymbol{F}, Q_t), T_{s+t}). \quad (11)$$

The pre-alignment similarity loss incentives generation of non-trivial encodings by maximizing the encoding difference before the alignment transformation.

In order to increase feature density, we introduce *best sample similarity* error. We first divide the encodings into a set of non-overlapping chunks, U. In our experiments, we used square chunks with width and height of 32 pixels. For each chunk, U_i, we compute the masked similarity error,

$$\psi(E(Q_s \circ \boldsymbol{F}, Q_t), T_{s+t} \cap U_i). \quad (12)$$

We rank the chunks by their similarity error, and select the bottom k-th percentile of chunks, V. We define *best sample similarity* as the average masked similarity error across the set of chunks V. The intuition is to select regions of the images that do not contain sufficient feature density to significantly contribute to the pre-alignment similarity loss. In our experiments, we set k to be the 50th percentile. Pre-alignment similarity can be formulated as

$$L_{pre} = -\frac{1}{|V|} \sum_{V_i \in V} \psi(E(Q_s, Q_t), T_{s+t} \cap V_i). \quad (13)$$

Finally, the stage 2 loss can be formulated as

$$L_2 = \alpha L_{pre} + L_{post} + \gamma L_{er}, \quad (14)$$

where α is a hyperparameter that balances pre-transformation and post-transformation similarity during training. In our experiments α was set to 0.7, and the second training stage lasted for 500 epochs.

After the end of the second training stage, the field is computed for the whole training dataset, and training proceeds to the next scale Aligner/Encoder pair.

Vector voting produces median displacements

We introduce vector voting, a function that smoothly computes the median displacement field from a set of fields,

$$\boldsymbol{F} = \nu(\{\boldsymbol{F}_1, \ldots, \boldsymbol{F}_n\}). \quad (15)$$

To start, we organize the n input fields into all combinations of subsets that constitute a minimum simple majority, such that

$$W = \binom{[1,\ldots,n]}{m}, \quad m = \left\lfloor \frac{n}{2} + 1 \right\rfloor. \quad (16)$$

For example, if $n = 3$, then $W = \{\{1,2\},\{1,3\},\{2,3\}\}$. We compute the average similarity between all pairs of fields for each subset, $W_k \in W$. In our experiments we used Euclidean distance,

$$D_k = \frac{1}{\binom{m}{2}} \sum_{i,j \in W_k, i \neq j} \| \boldsymbol{F}_i - \boldsymbol{F}_j \|. \quad (17)$$

We use a softmin function to map these average similarities to a normalized set of coefficients, such that

$$w_k = \frac{e^{-\frac{D_k}{T}}}{\sum_j e^{-\frac{D_j}{T}}}, \tag{18}$$

where T is the temperature. A lower softmin temperature, T, will give outlier vectors less weight in the output, but at a trade-off of potentially including undesirable discontinuities in the field as neighboring positions in the output may be averages from different subsets. For our experiments, we used $T = 5.7$ with $n = 3$. As an added insurance against introducing undesirable discontinuities, the fields may be spatially smoothed when computing their similarity in eq. (17).

For each subset, we distribute its coefficient evenly across its member fields, and the ultimate weight for each field in the input set is the sum of these distributed coefficients. The final field is then

$$\boldsymbol{F} = \sum_k \sum_{i \in W_k} \frac{w_k}{m} \boldsymbol{F}_i. \tag{19}$$

Vector voting corrects aligner errors

Given two images M_s and M_t as input, an aligner ϕ computes a displacement field

$$\phi(M_s, M_t) = \tilde{\boldsymbol{F}}_{s \to t}, \tag{20}$$

that transforms M_s to be similar to M_t. The subscripts s and t indicate section indices. The tilde sign indicates a field directly produced by an aligner, as opposed to a field that has undergone additional operations.

Aligners can use a previously computed field to transitively align a source to a previous target,

$$\phi(M_s, M_t \circ \boldsymbol{F}_{t \to k}) = \tilde{\boldsymbol{F}}_{s \to t \to k}, \tag{21}$$

as well as align a transformed source to a target,

$$\phi(M_s \circ \boldsymbol{F}_{s \to k}, M_t) = \tilde{\boldsymbol{F}}_{s \to k \to t}. \tag{22}$$

The field, $\tilde{\boldsymbol{F}}_{s \to t}$, can contain errors if the target image contains missing or corrupted information. If we have a set of images, $\{M_1, \ldots, M_n\}$, we can make multiple measurements of the same displacement field using transitive alignment. Now we can use vector voting as an error correction procedure with the set of displacement fields,

$$\boldsymbol{F}_{s \to t} = \nu(\tilde{\boldsymbol{F}}_{s \to 1 \to t}, \cdots, \tilde{\boldsymbol{F}}_{s \to n \to t}). \tag{23}$$

This assumes the Anna Karenina principle: correct transitive fields are relatively similar with other correct fields, while incorrect fields will be different in different ways. The smooth consensus of vector voting preserves desired smooth and non-smooth features of the set of displacement fields.

Sequential alignment

With an error correction procedure in place, we can now define sequential alignment. It starts by fixing the first section of the block, so that its field is the identity (section indices relative to the first section),

$$\boldsymbol{F}_0 = \boldsymbol{I}. \tag{24}$$

To provide multiple targets for voting on the next few sections in the sequence, we align a few previous sections to the first section,

$$\boldsymbol{F}_{-k \to 0} = \phi(M_{-k}, M_0), \tag{25}$$

where $k \in \{1, \ldots, n-1\}$. From multiple measurements of the displacement field,

$$\tilde{\boldsymbol{F}}_{i \to i-k \to 0} = \phi(M_i, M_{i-k} \circ \boldsymbol{F}_{i-k \to 0}) \tag{26}$$

we use voting to produce the final consensus field that aligns a given section to the start of the block,

$$\boldsymbol{F}_{i \to 0} = \nu\left(\{\tilde{\boldsymbol{F}}_{i \to i-k \to 0}\}_{k \in \{1, \ldots, n\}}\right). \tag{27}$$

Block Alignment

Starter sections, B, were manually selected by reviewing every tenth section at 1024 nm resolution, and identifying the nearest section that was deemed free of large defects. Block sizes were adjusted accordingly. Section indices are relative to the first section of the entire dataset.

The displacement between a neighboring pair of blocks, or stitching field, was computed by registering the overlapping n sections of the two blocks, $S = \{B_i + k : k \in \{0, \ldots, n-1\}\}$, then voting over the set of n fields that were produced.

$$\tilde{\boldsymbol{F}}_{j \to B_i \to j \to B_{i-1}} = \phi\left(M_j \circ \boldsymbol{F}_{j \to B_i}, M_j \circ \boldsymbol{F}_{j \to B_{i-1}}\right) \tag{28}$$

$$\boldsymbol{F}_{B_i \to B_{i-1}} = \nu\left(\{\tilde{\boldsymbol{F}}_{j \to B_i \to j \to B_{i-1}}\}_{j \in S}\right) \tag{29}$$

Blocks were stitched together by composing the displacement field of a section computed during block alignment with preceding stitching fields, such that

$$\boldsymbol{F}_z = \boldsymbol{F}_{z \to B_i} \circ \alpha\left(\boldsymbol{F}_{B_i \to B_{k-1}}, z - B_i\right) \circ \cdots \circ \alpha\left(\boldsymbol{F}_{B_1 \to B_0}, z - B_1\right). \tag{30}$$

The stitching fields are adjusted, or "decayed", based on their distance from z. For a distance n, a stitching field is decayed such that

$$\alpha(\boldsymbol{F}(\vec{r}), n) = \vec{r} + \beta(n) \sum_{\vec{x}} G_{\sigma(n)}(\vec{x}) \boldsymbol{u}(\vec{r} - \vec{x}), \tag{31}$$

where $G_{\sigma(n)}$ is a blurring kernel with varying standard deviation that reduces the high-frequency components of the field with distance, and $\beta(n)$ is a function that returns a coefficient to dampen the effect of the field with distance. In our experiments, we used

$$\sigma(n) = cn \text{ and } \beta(n) = \max(1 - \frac{n}{d}, 0), \tag{32}$$

with $c = 0.2$ (measured in pixels) and $d = 200$ (measured in sections). The effect of applying a varying blurring kernel on each stitching field was generated by trilinearly interpolating a MIP hierarchy of the field generated using an iterative 2×2 box filter[38].

Composition with decay preserves pair alignment

Block alignment should preserve pair alignment, such that,

$$\boldsymbol{F}_z \circ \boldsymbol{F}_{z-1}^{-1} \approx \boldsymbol{F}_{z \to z-1}. \tag{33}$$

For example, let's use fields that are in the first block, so that

$$\boldsymbol{F}_{z\to B_1} \circ \alpha\left(\boldsymbol{F}_{B_1\to B_0}, z - B_1\right) \circ \left[\boldsymbol{F}_{z-1\to B_1} \circ \alpha\left(\boldsymbol{F}_{B_1\to B_0}, z - 1 - B_1\right)\right]^{-1}. \tag{34}$$

The inverse of a composition is the composition of the inverses reversed, and an inverted aligner field aligns a target to its source,

$$\boldsymbol{F}_{z\to B_1} \circ \alpha\left(\boldsymbol{F}_{B_1\to B_0}, z - B_1\right) \circ \alpha\left(\boldsymbol{F}_{B_1\to B_0}, z - 1 - B_1\right)^{-1} \circ \boldsymbol{F}_{B_1\to z-1}. \tag{35}$$

We want to show that the second and third fields are roughly inverses,

$$\alpha\left(\boldsymbol{F}_{B_1\to B_0}, z - B_1\right) \circ \alpha\left(\boldsymbol{F}_{B_1\to B_0}, z - 1 - B_1\right)^{-1} \approx \boldsymbol{I}. \tag{36}$$

To simplify, let's set $n = z - B_1$. Starting with the vector equation for field decay, eq. (31), let's define a separate variable for the adjusted displacement,

$$\boldsymbol{U}_n(\vec{r}) = \beta(n) \sum_{\vec{x}} G_{\sigma(n)}(\vec{x}) \boldsymbol{u}(\vec{r} - \vec{x}). \tag{37}$$

This lets us write the inverse of the adjusted displacements as,

$$\boldsymbol{V}_n(\vec{r}) = -\boldsymbol{U}_n(\vec{r} + \boldsymbol{V}_n(\vec{r})). \tag{38}$$

Now we can rewrite eq. (36) as a vector equation,

$$\alpha\left(\boldsymbol{F}_{B_1\to B_0}, n\right) \circ \alpha\left(\boldsymbol{F}_{B_1\to B_0}, n - 1\right)^{-1} \tag{39}$$

$$= \alpha\left(\boldsymbol{F}_{B_1\to B_0}(\boldsymbol{r} + \boldsymbol{V}_{n-1}(\vec{r})), n\right) \tag{40}$$

$$= \vec{r} + \boldsymbol{V}_{n-1}(\vec{r}) + \boldsymbol{U}_n\left(\vec{r} + \boldsymbol{V}_{n-1}(\vec{r})\right) \tag{41}$$

$$= \vec{r} - \boldsymbol{U}_{n-1}\left(\vec{r} + \boldsymbol{V}_{n-1}(\vec{r})\right) + \boldsymbol{U}_n\left(\vec{r} + \boldsymbol{V}_{n-1}(\vec{r})\right) \tag{42}$$

Let $\vec{s} = \vec{r} + \boldsymbol{V}_{n-1}(\vec{r})$, and substitute the definition of the adjusted displacements,

$$\vec{r} - \beta(n-1) \sum_{\vec{x}} G_{\sigma(n-1)}(\vec{x}) \boldsymbol{u}(\vec{s} - \vec{x}) + \beta(n) \sum_{\vec{x}} G_{\sigma(n)}(\vec{x}) \boldsymbol{u}(\vec{s} - \vec{x}) \tag{43}$$

$$\vec{r} + \sum_{\vec{x}} \left[\beta(n) G_{\sigma(n)}(\vec{x}) - \beta(n-1) G_{\sigma(n-1)}(\vec{x})\right] \boldsymbol{u}(\vec{s} - \vec{x}) \tag{44}$$

If we use a very large decay distance so that $\beta(n) \approx 1$, then the filter convolving the displacements, $\boldsymbol{u}(\vec{x})$, is the difference between two blurring kernels with slightly different standard deviations. The slower that σ increases, the more similar the two blurring kernels, and the closer to producing the identity. Consider using a normalized Gaussian blurring kernel for G with a σ that increases at a rate of 0.02 px per section. This will see that the kernels do not differ to start, increasing to a peak difference of around 5% at about 22 sections, then decreasing to within 1% after 40 sections.

Large scale inference

In order to efficiently apply the proposed approach to large scale datasets, we designed an open source inference framework corgie (COnnectomics Registration Generalizable Inference Engine). The main functionalities of corgie are handling the dependencies between various steps in the pipeline and handling arbitrarily large images, while maintaining high utilization of distributed workers. Ability to gracefully handle dependencies is a major difference between corgie and other large scale 3D stack processing frameworks[39]. corgie is able to process arbitrarily large images by breaking them into smaller chunks, processing the chunks independently, and assembling the final result from the processed chunk outputs. As quality of convolutional network predictions tends to deteriorate near image boundaries, corgie allows the input chunks to overlap, combining their results either through cropping or soft blending. Processing of an individual chunk constitutes an atomic task, which will be assigned to one of the distributed workers. Instead of creating a static computation graph which encodes all of the dependencies between tasks, which may introduce overhead when the number of tasks is large, corgie uses dynamic task dispatch. More specifically, each processing step is defined as a Job. Upon being called, a job will yield either a set of tasks or a Barrier dependency, indicating that all of the tasks previously yielded by this job must be completed before the job can proceed.

As an example, alignment of each block will be executed as an independent block-align job. Block-align job is implemented as a combination of other jobs, such as compute-field job, render-job, vector-vote-job, etc.

corgie's task scheduler will distribute tasks from all of the simultaneously running jobs to the same pool of workers, track Barriers and task completion statuses, and schedule new tasks only when the job is ready to proceed. The corgie abstraction allows combining these jobs in straightforward, imperative way, while keeping cluster utilization high. corgie supports all of the processing steps described in this paper, including image pre-processing (normalization, defect mask computation and burn-in).

Training code

The metroem package implements the training code necessary for all of the models presented in this work, and is based on the PyTorch deep learning framework. The package is easy to install and operate, allowing users to train new models and finetune existing models on new data without extensive expertise in deep learning. Models trained through metroem can be used in corgie without any additional conversion, although corgie is not limited to using only metroem models. Additionally, metroem lets the users train aligners for multiple resolution with a single command.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

FAFB v15 data is publicly available from https://seung-lab.github.io/fafbv15/. The FAFB v14.1 dataset is hosted by BossDB[40] at https://bossdb.org/project/flywire. The two mouse cortex volumes are also hosted by BossDB at https://bossdb.org/project/microns-minnieand https://bossdb.org/project/microns-interneuron.

Code availability

Code is available under the Apache License 2.0 from https://github.com/seung-lab/corgieand https://github.com/seung-lab/metroem.

References

1. Harris, K. M. et al. Uniform serial sectioning for transmission electron microscopy. *J. Neurosci.* **26**, 12101–12103 (2006).
2. Briggman, K. L. & Bock, D. D. Volume electron microscopy for neuronal circuit reconstruction. *Curr. Opin. Neurobiol.* **22**, 154–161 (2012).

3. Zheng, Z. et al. A complete electron microscopy volume of the brain of adult drosophila melanogaster. *Cell* **174**, 730–743.e22 (2018).
4. Yin, W. et al. A petascale automated imaging pipeline for mapping neuronal circuits with high-throughput transmission electron microscopy. *Nat. Commun.* **11**, 4949 (2020).
5. MICrONS Consortium et al. Functional connectomics spanning multiple areas of mouse visual cortex. *bioRxiv* https://doi.org/10.1101/2021.07.28.454025 (2021).
6. Shapson-Coe, A. et al. A connectomic study of a petascale fragment of human cerebral cortex. *bioRxiv*https://doi.org/10.1101/2021.05.29.446289 (2021).
7. Lee, K. et al. Convolutional nets for reconstructing neural circuits from brain images acquired by serial section electron microscopy. *Curr. Opin. Neurobiol.* **55**, 188–198 (2019).
8. Khairy, K., Denisov, G. & Saalfeld, S. Joint deformable registration of large EM image volumes: a matrix solver approach. *arXiv* https://arxiv.org/abs/1804.10019 (2018).
9. Saalfeld, S., Fetter, R., Cardona, A. & Tomancak, P. Elastic volume reconstruction from series of ultra-thin microscopy sections. *Nat. Methods* **9**, 717 (2012).
10. Bock, D. D. et al. Network anatomy and in vivo physiology of visual cortical neurons. *Nature* **471**, 177–182 (2011).
11. Wetzel, A. W. et al. Registering large volume serial-section electron microscopy image sets for neural circuit reconstruction using FFT signal whitening. In *2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)* 1–10 (IEEE, 2016).
12. Scheffer, L. K., Karsh, B. & Vitaladevun, S. Automated alignment of imperfect EM images for neural reconstruction. *arXiv* https://arxiv.org/abs/1304.6034 (2013).
13. Lowe, D. G. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision* 1150–1157 (IEEE, 1999).
14. Kazhdan, M. et al. Gradient-domain fusion for color correction in large EM image stacks. *arXiv* https://arxiv.org/abs/1506.02079 (2015).
15. Turner, N. L. et al. Reconstruction of neocortex: Organelles, compartments, cells, circuits, and activity. *Cell* **185**, 1082–1100 (2022).
16. Mitchell, E., Keselj, S., Popovych, S., Buniatyan, D. & Seung, H. S. Siamese encoding and alignment by multiscale learning with self-supervision. *arXiv* https://arxiv.org/abs/1904.02643 (2019).
17. Yoo, I. et al. ssEMnet: Serial-section electron microscopy image registration using a spatial transformer network with learned features. *Deep Learn.* https://doi.org/10.1007/978-3-319-67558-9_29 (2017).
18. Balakrishnan, G., Zhao, A., Sabuncu, M. R., Guttag, J. & Dalca, A. V. VoxelMorph: A learning framework for deformable medical image registration. *IEEE Trans. Med. Imag.* https://ieeexplore.ieee.org/document/8633930 (2019).
19. Zhou, S. et al. Fast and accurate electron microscopy image registration with 3D convolution. In *Medical Image Computing and Computer Assisted Intervention (MICCAI)* 478–486 (Springer International Publishing, 2019).
20. Shu, C., Xin, T., Zhou, F., Chen, X. & Han, H. Dual networks for High-Precision and High-Speed registration of brain electron microscopy images. *Brain Sci.* **10**, 453312 (2020).
21. Nguyen-Duc, T. et al. Weakly supervised learning in deformable EM image registration using slice interpolation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI)* 670–673 (ISBI, 2019).
22. Jain, V. Adversarial image alignment and interpolation. *arXiv* https://arxiv.org/abs/1707.00067 (2017).
23. Popovych, S., Alexander Bae, J. & Seung, H. S. Caesar: Segment-wise alignment method for solving discontinuous deformations. In *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)* 1214–1218 (IEEE, 2020).
24. Bajcsy, R. & Kovačič, S. Multiresolution elastic matching. *Comp. Vision Graph. Image Process.* **46**, 1–21 (1989).
25. Ranjan, A. & Black, M. J. Optical flow estimation using a spatial pyramid network. *arXiv* https://arxiv.org/abs/1611.00850 (2016).
26. Bromley, J., Guyon, I., LeCun, Y., Sackinger, E. & Shah, R. Signature verification using a "siamese" time delay neural network. In *Proceedings of the 6th International Conference on Neural Information Processing Systems* 737–744 (Morgan Kaufmann Publishers Inc. USA, 1993).
27. Broit, C. *Optimal Registration of Deformed Images*. Ph.D. Thesis (University of Pennsylvania, 1981).
28. Tasdizen, T. et al. Automatic mosaicking and volume assembly for high-throughput serial-section transmission electron microscopy. *J. Neurosci. Methods* **193**, 132–144 (2010).
29. Möller, B., Garcia, R. & Posch, S. Towards objective quality assessment of image registration results. In *VISAPP Proceedings of the Second International Conference on Computer Vision Theory and Applications* 233–242 (VISAPP, 2007).
30. Li, P. H. et al. Automated reconstruction of a serial-section em drosophila brain with flood-filling networks and local realignment. *bioRxiv* https://doi.org/10.1101/605634 (2020).
31. Dorkenwald, S. et al. FlyWire: online community for whole-brain connectomics. *Nat. Methods* **19**, 119–128 (2022).
32. Macrina, T. et al. Petascale neural circuit reconstruction: automated methods. *bioRxiv* https://doi.org/10.1101/2021.08.04.455162 (2021).
33. Hui, T.W., Tang, X. & Loy, C.C. LiteFlowNet: A lightweight convolutional neural network for optical flow estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 8981–8989 (IEEE, 2018).
34. Wang, Z., Simoncelli, E. P. & Bovik, A. C. Multiscale structural similarity for image quality assessment. *Thrity-Seven. Asilomar Confer. Sign. Syst. Comp.* **2**, 1398–1402 (2003).
35. Mahalingam, G. et al. A scalable and modular automated pipeline for stitching of large electron microscopy datasets. *eLife* **11**, e76534 (2022).
36. Buniatyan, D. et al. Weakly supervised deep metric learning for template matching. In *Advances in Computer Vision* 3rd edn, Vol. 2 (Springer International Publishing, 2020).
37. Jaderberg, M., Simonyan, K., Zisserman, A. & Kavukcuoglu, K. Spatial transformer networks. In *Advances in Neural Information Processing Systems* 5th edn, Vol. 6 (*eds.* Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M. & Garnett, R.) Ch. 2017–2025 (Curran Associates, Inc., 2015).
38. Williams, L. Pyramidal parametrics. In *Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '83 1–11 (Association for Computing Machinery USA, 1983).
39. Wu, J., Silversmith, W. M., Lee, K. & Seung, H. S. Chunkflow: hybrid cloud processing of large 3D images by convolutional nets. *Nat. Methods* https://doi.org/10.1038/s41592-021-01088-5 (2021).
40. Hider, R. et al. The brain observatory storage service and database (bossdb): a cloud-native approach for petascale neuroscience discovery. *Front. Neuroinform.* https://doi.org/10.3389/fninf.2022.828787 (2022).

Acknowledgements

The research was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DoI/IBC) contract numbers D16PC00003, D16PC00004, and D16PC0005. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. HSS also acknowledges support from NIH/NINDS U19 NS104648, NIH/NEI R01 EY027036, NIH/NIMH U01 MH114824, NIH/NINDS

R01NS104926, NIH/NIMH RF1MH117815, NIH/NIMH RF1MH123400 and the Mathers Foundation, as well as assistance from Google, Amazon, and Intel. These companies had no influence on the research. We are grateful for support with FAFB imagery from D. Bock through NIH/NINDS RF1MH120679, and software administrative support provided by Tom Kazimiers (Kazmos GmbH) and Eric Perlman (Yikes LLC). We are grateful for support with mouse cortex imagery from N. da Costa, R. Torres, G. Mahalingam, R.C. Reid. We thank A. Burke, J. Gager, J. Hebditch, S. Koolman, M. Moore, S. Morejohn, B. Silverman, K. Willie, R. Willie for their image analyses, S-C. Yu for managing image analysis, G. McGrath for computer system administration, and M. Husseini and L. and J. Jackel for project administration. We are grateful to J. Maitin-Shepard for making Neuroglancer freely available. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/IBC, or the U.S. Government.

Author contributions

E.T.T. and S.S. revised montages and rigidly aligned FAFB v15. J.A.B. detected cracks and folds within FAFB v15. N.K. detected tissue within FAFB v15. N.K., S.P. coarse aligned FAFB v15. N.K., S.P., and T.M. fine aligned FAFB v15. S.P. aligned the mouse cortex sample. S.P., with help from T.M., developed metroem. S.P., with help from T.M., N.K., and M.C., developed Corgie. E.M., S.P., with contributions from T.M., N.K., M.C., B.N., Z.J., J.A.B., S.M., and K.L., developed the preliminary pipeline. S.P., M.C., N.K., and T.M. aligned the mouse cortex in MICrONS release. S.P., T.M., and H.S.S. wrote the manuscript. H.S.S. led the project.

Competing interests

The following authors declare the following competing interests: H.S.S., N.K., S.P., and T.M. disclose financial interests in Zetta AI LLC. The remaining authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41467-023-44354-0.

Correspondence and requests for materials should be addressed to H. Sebastian Seung.

Peer review information *Nature Communications* thanks Yoshiyuki Kubota, and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.