Enhancement of brain atlases with region-specific coordinate systems: flatmaps and barrel column annotations

Sirio Bolaños-Puchet[1,†], Aleksandra Teska[1,2,†], and Michael W. Reimann[1,*]

[1]Blue Brain Project, École polytechnique fédérale de Lausanne (EPFL), Campus Biotech, 1202 Geneva, Switzerland
[2]Laboratory of Sensory Processing, Brain Mind Institute, Faculty of Life Sciences, École polytechnique fédérale de Lausanne (EPFL), Switzerland
[†]Co-lead authors
[*]Corresponding authors

August 24, 2023

Abstract

Digital brain atlases define a hierarchy of brain regions and their locations in three-dimensional space. They provide a standard coordinate system in which diverse datasets can be integrated for visualization and analysis. They also enable building of data-driven computational models of brain regions. For atlases of the cerebral cortex, additional information is required to work effectively with its particular, layered architecture and curved geometry. Although some approaches have been employed in the literature, no usable method to produce such information is openly available. To fill this gap, we describe here methods to enhance a cortical atlas with three auxiliary, voxel-wise datasets: first, a field of cortical depth; second, a field of local orientations towards the cortical surface; and third, a *flatmap* of the cortical volume: a two-dimensional map where each pixel represents a subvolume of voxels along the depth axis, akin to a cortical column. We apply these methods to the somatosensory regions of a digitized version of Paxinos and Watson's rat brain atlas, and define metrics to assess the quality of our results. Among the many applications of the resulting flatmap, we show their usefulness for: decomposing the cortical volume into uniform columnar subvolumes and defining a topographic mapping for long-range connections between subregions. We also generate a flatmap of the isocortex regions of the Allen Mouse Common Coordinate Framework. Combining this with established two-photon tomography data, we then annotate individual barrels and barrel columns in the mouse barrel cortex. Finally, we use the flatmap to visualize volumetric data and long-range axons. We provide an open source implementation of our methods for the benefit of the community.

1 Introduction

Experimental and computational studies of the rodent brain make increasing use of standardized atlases as reference spaces in which to integrate and analyze data, which can be used as the building blocks in data-driven modeling. Atlases providing annotations of cortical and subcortical structures are available for both rats (Paxinos and Watson, 2007; Papp et al., 2014) and mice (Paxinos and Franklin, 2019; Wang et al., 2020).

While traditional printed atlases consist of a series of labeled outlines of brain slices along different cut planes, digital brain atlases represent information as a three-dimensional matrix where each voxel is associated with the region to which it belongs. Typically, the Cartesian coordinate system defined by the voxel indices is aligned to match the anatomical axes of the brain (transverse, coronal, sagittal). As such, digital brain atlases provide a lookup for the locations of brain regions in global anatomical coordinates.

Although the Cartesian coordinate system is anatomically well-defined, it does not provide the best context in which to describe the shape and structure of brain regions. Many regions have been traditionally analyzed and modeled with respect to region-specific coordinate systems that better capture their functional and developmental organization. For example, hippocampus has a complex geometry where a path following its local depth axis tends to be curved, and different paths can have entirely opposite orientations in the different hippocampal subfields. Since these properties cannot be accounted for by simple linear or affine transforms of the Cartesian coordinate system, hippocampal geometry has been described in terms of long-axis, proximal-distal and laminar coordinates in human (DeKraker et al., 2018) or longitudinal, transverse and radial coordinates in rat (Romani et al., 2023). Even thalamic nuclei contain functional gradients that can be expressed as axes of a region-specific coordinate system (Sumser et al., 2017).

Furthermore, as data is recorded over increasingly larger regions, the lack of continuous region-wide coordinate systems may begin to pose a problem for data analysis. Also, the construction of large-scale models of entire brain regions is likely to rely on algorithms that are defined in such coordinate systems adapted to the particular geometry of each region. As such, it becomes imperative to provide a reproducible, objective way to define them.

In this manuscript, we focus on a particular kind of region-specific coordinate system, inspired by the use case of layered regions such as the neocortex. We introduce the following requirements for a *laminar coordinate system*. First, it is primarily defined by one *principal axis* of organization whose orientation varies smoothly in space. Second, the orientation of this principal axis can be largely determined by the direction along which the layers change at each location. Third, the other two axes are locally orthogonal to the principal axis, but otherwise their orientation is of secondary importance.

We present algorithms to define a laminar coordinate system based on user specifications, approximating as much as possible the conditions given above. To ensure flexibility, and given the discrete nature of the input data, we do not attempt to derive a mathematical description of the coordinate system. Instead, we represent it in the form of an *auxiliary atlas*, a set of three-dimensional matrices in the same Cartesian space as the base atlas, containing values for the local coordinates at the center of each voxel. Additionally, given its importance, we also provide the principal axis vector at each voxel. We call the process of generating auxiliary atlas datasets *atlas enhancement*.

Considering only the two non-principal axes, provides a two-dimensional representation of the three-dimensional cortical volume. Intuitively, this is akin to compressing the neocortex by pushing simultaneously from the top and bottom, collapsing all layers perfectly on top of each other. The result is a very thin, but curved sheet of cortex; by flattening this sheet, and drawing coordinate axes on it, we obtain what is called a *flatmap*. Such a representation has been obtained for mouse isocortex before (Wang et al., 2020). The main property of a flatmap is that it allows the seamless mapping of information back and forth between 2D and 3D spaces in a way consistent with the principal axis. For example, a circle drawn on the flatmap corresponds to a 3D subvolume similar to a cylinder, whose axis follows the local direction of the principal axis, and where the layer structure is preserved. In the context of the neocortex, such a subvolume would represent a cortical column, if appropriately sized. Other important properties of a flatmap are *connectedness*, i.e. the 3D subvolume associated with a 2D area is connected, and *continuity*, i.e. 3D subvolumes associated with neighboring areas are neighbors themselves. We measure these and other properties to assess the quality of our results.

We apply our methods to enhance an atlas of rat somatosensory cortex (based on Paxinos and Watson, 2007) with a full three-dimensional coordinate system adapted to this region. We also enhance the Allen Mouse Common Coordinate Framework version 3 (CCFv3, Wang et al., 2020) by generating a flatmap of the whole mouse isocortex and refined the annotations of the barrel cortex. Despite the barrel cortex being one of the most studied regions of the rodent brain, previous efforts on barrel segmentation have been made only in rat (Egger et al., 2012), and existing mouse atlases currently lack annotations of individual barrels (Welker and Woolsey, 1974; Simons and Woolsey, 1979). To address this, we segmented well-defined barrels from the average template dataset in CCFv3, and used our flatmap to annotate individual anatomical units and their associated barrel columns (Wang et al., 2020).

We further explore applications of flatmaps for modeling and data analysis and visualization. Finally, we provide an open source implementation of our algorithms for the benefit of the community.

2 Methods

2.1 Generating depth and orientation fields

The input to the algorithm is a voxelized digital brain atlas with region annotations, i.e., each voxel has a value representing the brain region to which it belongs. We select a subset of regions, e.g. all isocortex regions or somatosensory regions, and use them as the *source volume* for the following steps. Specifically, cortical depth information at each voxel will be represented as relative depth (between 0 and 1) and cortical thickness, from which absolute depth can be easily computed as the product of the two.

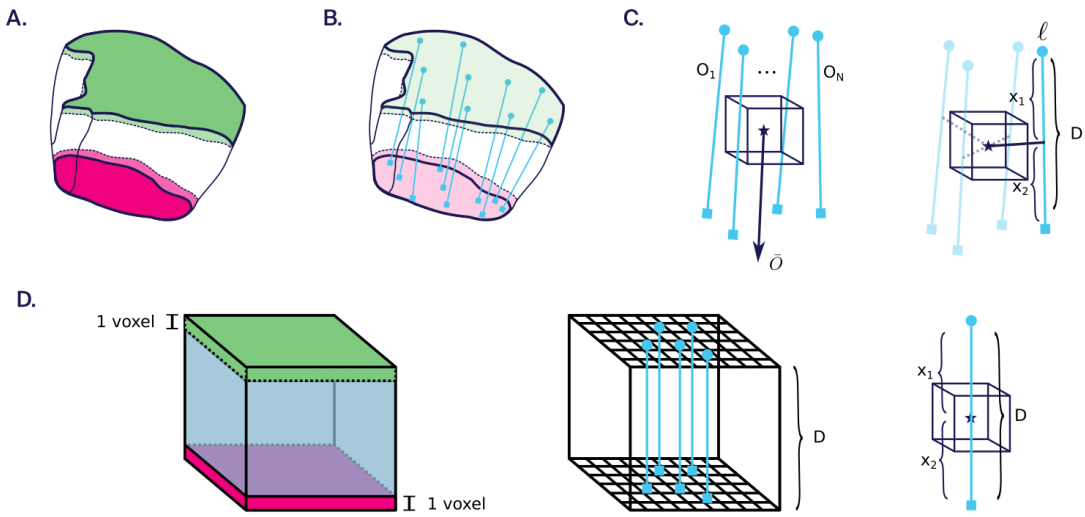

Figure 1: **Generating depth and orientation fields. A.** Top (green) and bottom (pink) shells. **B.** Shortest line segments (light blue) between pairs of voxels in top and bottom shells. **C.** Left: Average direction of N closest line segments gives local orientation $\bar{O}$. Right: Projection onto closest line segment ℓ gives local depth $d = \frac{x_1}{D}$. Length of closest line segment gives local thickness D. **D.** Same as before, but for the simple geometry of a cube as an ideal example. In addition to top and bottom shells one voxel thick, sides are labeled (light blue). Shortest line segments are all parallel and same height, and there is a single line segment passing through every voxel center.

We begin by segmenting the top and bottom voxel shells of the source volume. These shells have a thickness of one voxel, and are located at the boundary of the source volume; they represent zero depth and maximum depth, and correspond to the cortical surface and the white matter boundary, respectively (Fig. 1A). One possible strategy to select these shells would be to take all voxels that are in contact with the outside of the source volume. However, this is merely a required, but not a sufficient condition, as many of the voxels selected this way would fall on the "sides" of the source volume at different depths. This, and the following steps of the method, are illustrated for the simple geometry of a cube to get an intuition (Fig. 1D). Given the complexity of this task, this step is best performed by expert manual annotation to split top, bottom, and sides of the source volume.

Having done this, we proceed to find for each voxel in the top shell, the closest voxel in the bottom shell, and vice-versa. We define all line segments joining pairs of voxel centers and store them (Fig. 1B). We use this bundle of line segments to define the local orientation of the principal axis and the total extent of the source volume along the principal axis (i.e. cortical thickness) at each voxel (Fig. 1C).

To define the local orientation vector, we consider the N closest line segments to the voxel center and take their average direction $\bar{O}$ in the sense pointing towards the bottom shell. To define the local cortical thickness, we take the line segment ℓ closest to the voxel center and measure its length D. Finally, to define the local relative depth, we project the voxel center onto ℓ and take the ratio of the upper part (towards the top shell) to its total length $d = \frac{x_1}{D}$.

The number N of direction samples to use per voxel, determines to what degree the resulting orientation field is spatially smoothed. Artifacts in the region annotation atlas and its spatial discretization can lead to sudden jumps of the measurements, making some degree of smoothing necessary; so N should be chosen depending on voxel resolution and the geometry of the source volume. Being able to smooth the direction of the orientation vector is the reason why we use this procedure instead of just taking the gradient of the depth field as orientation field.

The result of this procedure are three atlas datasets that together define the principal axis in the source volume: two scalars for the total thickness and relative depth, and one vector for the local orientation. The orientation vector is represented as a rotation of the Cartesian Y axis, and the associated rotation matrix is stored on file as a quaternion.

2.2 Flatmapping

The input for the flatmapping stage of the algorithm is a definition of the principal axis, along which the volume is to be flattened. Specifically, we require voxelized datasets of local orientation and relative depth. They can be obtained using the methods described in the previous section, but also by other methods, e.g., solving the heat equation between "hot" and "cold" top/bottom shells with adiabatic sides (Jones et al., 2000; Airey et al., 2005; Lerch et al., 2008; Wang et al., 2020), and taking the solution itself as the depth field and its gradient as the local orientation.

In what follows, we define $d(x)$ as a function associating to any point x in the source volume a relative depth value through linear interpolation of the voxelized relative depth dataset, and correspondingly $O(x)$ for the orientation vector.

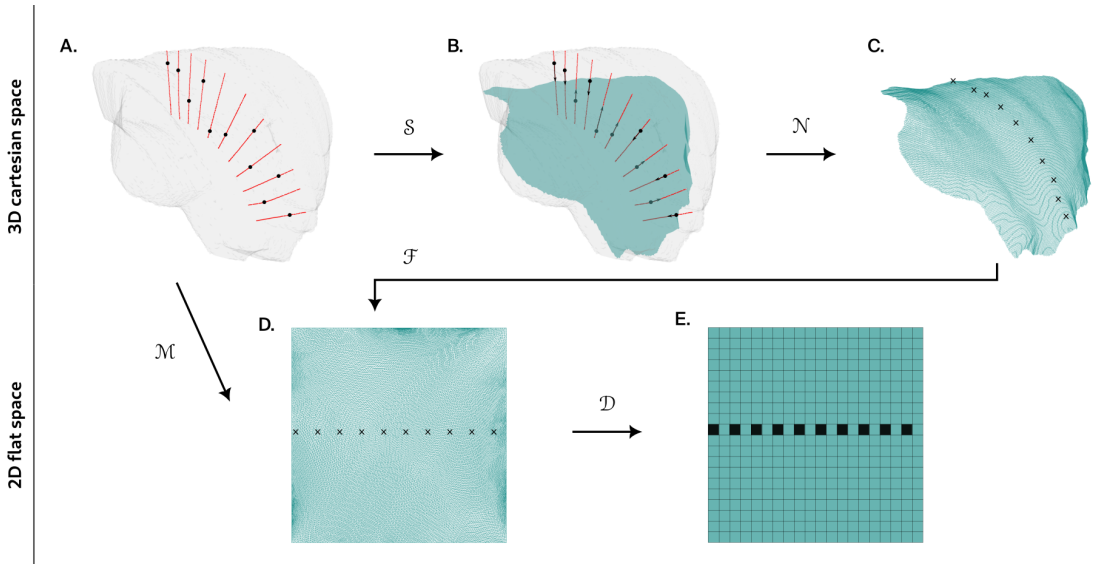

Figure 2: **Flatmapping algorithm. A.** Streamlines (red curves) are computed by numerical integration of the orientation field using voxel centers (black dots) as initial positions. **B.** Voxel centers are mapped to the projection surface (turquoise) following streamlines (black arrows, operator $\mathcal{S}$). **C.** Points on the projection surface are mapped to vertices (black crosses) on the reconstructed projection mesh (turquoise wireframe) through nearest-neighbor search (operator $\mathcal{N}$). **D.** Authalic (area-preserving) flattening of projection mesh into the unit square (operator $\mathcal{F}$). The composite flatmapping operator $\mathcal{M} = \mathcal{F}(\mathcal{N}(\mathcal{S}(\cdot)))$ maps 3D voxel centers to 2D flat coordinates. **E.** Flatmap discretization by grouping of mesh vertices into pixels of chosen size (operator $\mathcal{D}$).

Stage A.1 Define projection mesh The first step is choosing a *projection surface* onto which three-dimensional points are projected. This surface is defined as a level set (isosurface) of the relative depth field, i.e., the set of all points at the same relative depth d^*. Note that this surface is only defined as an abstraction, as it cannot be computed exactly in a numerical setting.

What can be defined is a *projection mesh*, a triangular mesh that approximates the projection surface. To this end, first we find all voxels with a relative depth close to d^* and define a point cloud with their centers. This is done by taking a mask consisting of a) all voxels outside the region, and b) with depth values at or above d^*, then compute the euclidean distance transform (EDT) and finally select voxels in (b) with an EDT value of 1. Then, we apply a surface reconstruction algorithm to generate a triangular mesh from the point cloud. While this is a well-known subject in the field of computational geometry, with many available approaches, here we used specifically the scale space surface reconstruction algorithm (van Lankveld, 2023). This algorithm gave the best results for our purposes, as it produces a mesh that contains the original points and thus approximately matches the projection surface.

Finally, we refine the triangular mesh. Since we are going to map voxel centers to vertices in this mesh, we wish to avoid discretization errors from having too few vertices. Furthermore, the issue is not only to have enough vertices, but to have them distributed everywhere in the mesh. We find that this can be achieved efficiently through refinement by subdivision (Geuzaine and Remacle, 2009), that is, introducing new vertices at the midpoints between existing vertices and connecting them to define new triangles.

Stage A.2: Flatten projection mesh Flattening of the projection mesh is performed using an authalic (area-preserving) mapping algorithm (Saboret et al., 2023) that maps the mesh boundary to the unit square. The algorithm takes as input an open triangular mesh (i.e., having a boundary) with a single connected component, and outputs a triangular mesh that is flat, i.e., has only X and Y coordinates in the range $[0, 1]$, with the same number of points as the input mesh in a one-to-one correspondence. An area-preserving algorithm is chosen to ensure that the relative sizes of the regions in the source volume is preserved. We call this flattened triangular mesh in $[0, 1] \times [0, 1]$ the *flat space* and $\mathcal{F}$ is the flattening operator that maps points on the projection mesh to points in flat space.

Stage B.1: Compute streamlines In parallel to stage A and independent from it, we compute a mapping from voxel centers to points on the projection mesh (Fig. 2A).

For each voxel v in the source volume we generate a *streamline*, a parameterized 3D curve that follows the principal axis at every point and connects the top and bottom shells while passing through the voxel center. In detail, with the center of v as initial position, we compute an integral curve of the orientation field O using a numerical differential equation solver. The integration step of the solver is set small enough to have gradual change in depth, which results in very smooth curves. First we iterate in the forward direction until we reach a relative depth of 1 (with epsilon tolerance); then, we iterate in the backward direction until we reach a relative depth of 0. Finally, we join these two curve segments into a parameterized curve $S_v(t)$ associated with voxel v, where the parameter $t \in [0, 1]$ maps to points along the curve with linear spline interpolation.

Stage B.2: Project voxel centers We use the streamline $S_v(t)$ to find the projection of voxel v onto the projection surface (Fig. 2B). Intuitively, we move the voxel center along its streamline until reaching the projection surface. Technically, we define $F(t) = d(S_v(t)) - d^*$ and use a root finding algorithm to find t_v^*, the point where the function becomes zero and the point along the streamline is at relative depth d^*. We call $\mathcal{S} = S_v(t_v^*)$ the operator that maps voxel centers to points on the projection surface.

For some voxels, especially at the edges of the source volume, integration of O may not reach relative depths of 0 or 1 before leaving the volume. This may happen due to noise or discontinuities in the orientation field, or truncation of the geometry. These voxels are not *eligible* for flattening and we discard them, since they may not have a projection (if their streamline never attains depth d^*), and even if they do, the laminar architecture is not preserved along their streamlines.

Stage B.3: Map streamline intersections to mesh points Once we have the projections of voxel centers onto the projection surface, we use a nearest neighbor search (Tangelder and Fabri, 2023) to find the corresponding closest points on the projection mesh. We call this operator $\mathcal{N}$ (Fig. 2C).

This step demonstrates the need to refine the projection mesh. While it is expected that points along the same or very close streamlines are mapped to the same mesh vertex by $\mathcal{N}$, if this happens for two distant voxel centers, it results in a discretization error. In principle, this could be avoided entirely if we added a mesh vertex at the location of every streamline intersection. However, by applying enough refinement steps we manage to reduce the discretization errors to a negligible level. We find in practice that a number of vertices equal to one third or more of the number of voxels in the source volume works well.

Final step: Flatmap generation and discretization By combining the outputs from previous stages, we are able to map the center of every voxel v in the source volume to a point in flat space, its *flat projection*, via the composite *flatmap* operator $\mathcal{M} = \mathcal{F} \circ \mathcal{N} \circ \mathcal{S}$. The coordinates of the flat projection are called the *flat coordinates* of the voxel. Note that these have an intrinsic maximum resolution given by the number of vertices in the projection mesh; we call this the *native resolution* of the flatmap.

For practical purposes, it is desirable to have a regular discretization of the flat space as a 2D array with *pixel resolution* N, meaning that floating-point coordinates in $[0, 1]$ are binned into integer indices from 0 to $N - 1$ in both axes (Fig 2E). This decrease in resolution can be advantageous, depending on the use case (see Results).

In general, many voxels will have the same flat projection, and the set of voxels that map to a given point in flat space, whether at native or pixel resolution, is called the *pre-image* of that point (inverse operator $\mathcal{M}^{-1}$). Pixels with empty pre-images, i.e. that no voxels are mapped to, may be undesirable depending on the application. In that case, there is a maximum resolution that avoids the formation of such holes in flat space (see Results).

2.3 Distance and area measurements in flat space

Given that the mesh flattening algorithm always maps the projection mesh to the unit square, flat coordinates are normalized to $[0, 1]$ and distance units are lost. In order to recover distance units in flat space, we approximate the size of pre-images (in μm) at a given pixel resolution (see below) and then establish a linear fit between mean pre-image size and pixel size. The slope of the linear fit can be used as proportionality factor between distances in flat space and Cartesian space.

To measure areas in flat space, since the flattening algorithm is area-preserving, it suffices to use the original area of the projection mesh as proportionality factor.

2.4 Characterization of depth and orientation fields

In order to quantify the smoothness of the resulting depth and orientation fields, we compute the Laplacian of the former and the divergence of the latter at every voxel, and check how close these quantities are to zero, i.e., how close the depth and orientation fields are to being a homogeneous solution to the heat equation and its gradient, respectively. We compute statistics of these quantities over all interior voxels (not in the top, bottom or sides), to reduce bias due to discrete numerical derivative artifacts at the boundaries of the domain.

2.5 Flatmap characterization

As mentioned before, important properties of a flatmap are: continuity, connectedness, and mapping of 2D areas to 3D subvolumes with intact layer structure. We present some metrics to characterize how close is the output of our flatmapping algorithm to having these desired properties. There are some global or per-voxel metrics, but otherwise they are defined for each pixel at some flat resolution. These per-pixel metrics characterize the structure of the pre-images and can be further analyzed in terms of statistics for all pixels computed across pixel resolutions.

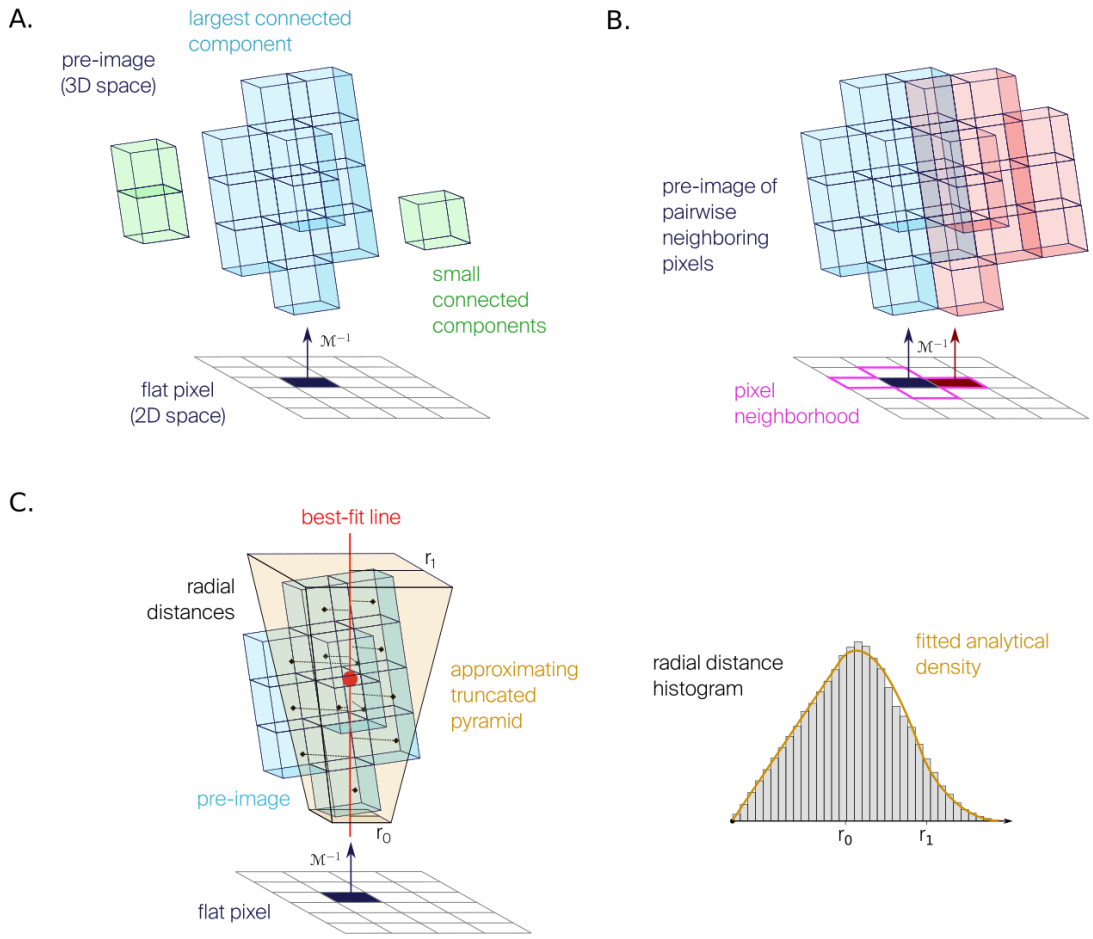

Figure 3: **Flatmap characterization. A.** Pre-image size and connectedness. The pre-image may consist of more than one connected component. **B.** Pre-image continuity for a pair of neighboring pixels (red, blue). The metric is evaluated over the 4 pairs in the pixel neighborhood (magenta). **C.** Pre-image radius and conicity are computed from the radii of the best approximating truncated square pyramid. The analytical density is fitted to the histogram of radial distances from the best-fit line.

2.5.1 Global metrics

Coverage. Fraction of eligible voxels in the source volume, i.e., having a flat projection. A value lower than the fraction of interior voxels in the source volume could indicate an issue with the orientation field, since issues of incomplete streamlines are only expected at the edges.

Usage fraction. Fraction of pixels in flat space with a non-empty pre-image, computed at some pixel resolution. At low resolution the usage fraction is almost certainly equal to 1, whereas at higher resolutions holes eventually appear and the usage fraction becomes lower than 1. The right pixel resolution depends on the application, and this metric can help in choosing it. At native resolution, the usage fraction is the ratio of the number of vertices with non-zero pre-images to the total number of vertices in the mesh.

2.5.2 Per-voxel metrics

Orthogonality. Degree to which the principal axis is orthogonal to the flatmap axes at each voxel. We compute this by first computing the gradient of each flat coordinate taken as a voxel dataset in Cartesian space. Then, we compute the cross product of these two gradients and choose the sense towards the bottom shell. Finally, we measure the cosine of the angle between this vector and the orientation vector by taking the dot product after normalization. A value of 1 indicates that the principal axis is locally orthogonal to the flatmap, whereas a lower value indicates some degree of non-orthogonality. If this quantity is negative it means the cross product points in the opposite direction as the orientation vector, and a sign flip is needed for proper comparison.

2.5.3 Per-pixel metrics

Pre-image size uniformity. Number of voxels in the pre-image of a pixel, normalized by the total number of eligible voxels, and compared to the fraction expected from pixel area in the perfectly uniform case. In the ideal case, this value would reflect the purely geometrical variations in thickness of the source volume, as this would determine how many voxels project to this pixel while following streamlines locally. However, noise and artifacts in the orientation field can lead to off-site streamlines that artificially increase the pre-image size at locations far from the neighborhood of a voxel, leading to non-uniformity.

Pre-image radius. Extent of the pre-image of a pixel in the directions orthogonal to its main axis. To give some intuition, if the pre-image was a cylinder, this would be its radius. However, for a square pixel the pre-image looks more like a truncated square pyramid, so we require a way to measure the extent of such an object. We approximate it as follows. The center of the pre-image is calculated as the centroid of its voxels. The main axis of the pre-image is defined as a line passing through its center and having the direction that minimizes the sum of square distances to all of its voxels, i.e., the first principal component of its voxel coordinates. Then, the distance from the main axis to each voxel $r(v)$ is calculated, and its distribution is compared to the analytical result for a truncated square pyramid of radii r_0 and r_1 ($r_0 \leq r_1$) (Fig. 3C). The values of r_0 and r_1 are then optimized to best fit the empirical distribution function of $r(v)$. Finally, we define the pre-image radius as $(r_0+r_1)/2$.

Pre-image conicity. Degree to which the pre-image of a pixel resembles a cone. This is quantified with one minus the quotient of the smaller to the larger radius of the truncated square pyramid fitted to the pre-image: 1 - r_0/r_1. This quantity is equal to 0 for a right square prism (no conicity), and has values closer to 1 the steeper the pyramid is.

Pre-image connectedness. Ratio of the number of voxels in the largest connected component of the pre-image of a pixel (Fig. 3A, light blue), to the number of voxels in the entire pre-image (Fig. 3A, light blue and green). We consider a pair of voxels to be connected if one is in the 3D Moore neighborhood (26 adjacent voxels) of the other. This metric quantifies how much the pre-image consists of a single chunk vs multiple separate pieces. By taking the ratio of the largest component to the total, instead of the number of connected components, we avoid bias due to possibly one large and many small components (even single voxels). If the flat resolution is too high, chances are that pre-images will become disconnected as the noise in the geometry and the orientation field becomes more evident.

Pairwise pre-image continuity. Ratio of the number of voxels in the largest connected component of the pre-image of a pair of neighboring pixels (Fig. 3B, blue and red), to the number of voxels in the entire pre-image, taking the minimum (worse case) over pairs in all four cardinal directions (Fig. 3B, magenta). This metric provides a measure of local continuity in the sense that going from one pixel to the next in flat space represents a movement between contiguous places in Cartesian space. This is important, as otherwise it would not be possible to relate close points in Cartesian space to close points in flat space. A discontinuous flatmap would indicate issues with the orientation field, as streamlines should vary smoothly and not cross or deviate from their neighbors.

2.6 Applications

2.6.1 Volume decomposition

The flatmap can be used to decompose a cortical volume into uniform and non-overlapping subvolumes, each sampling the complete stack of cortical layers. To define such a subvolume, first we draw a shape in flat space and take all points (at native resolution) that are inside this shape. Then, we take the pre-images of all these points and consider their union, i.e., the set of all voxels that under the flatmap operator project to this flat shape. The resulting set of voxels describes a 3D volume that "follows" the streamlines and keeps the layer structure intact. If we draw a tiling of flat space (square or hexagonal grid, etc.) and consider the subvolumes corresponding to all elements of the tiling or grid, we obtain a decomposition of the entire cortical volume. The subvolumes are non-overlapping as long as the flat shapes are non-overlapping, as is the case for a plane tiling, and their properties can be characterized using the metrics described above (taking "per-pixel" to mean per element of the tiling or grid).

2.6.2 Annotation of barrel columns

A variant of the volume decomposition procedure can be used to generate annotations of barrel columns. For that use case, the shapes drawn in flat space are the outlines of the individual barrels, generated as follows. Firstly, barrel annotations were extracted based on the "average template" two-photon tomography dataset part of CCFv3 (Wang et al., 2020), where the anatomical features of the barrel and septum separation are salient. Secondly, we performed a semi-automatic segmentation in ITK-SNAP, with the use of the Active Contour Snake algorithm (Yushkevich et al., 2016), and obtained volumes for 33 barrels in the left hemisphere. We annotated the 33 barrels as separate structures and labeled them with their corresponding rows and arcs. Thirdly, the new annotations were projected onto flat space (Fig 6A). We used the flat positions to further refine the segmentation by reassigning mislabeled voxels to the label of their nearest neighbor. Lastly, with the final segmentation we computed the convex hull of all flat positions belonging to a barrel, and took the pre-image of all points inside it to obtain barrel columns (Fig 6B and (Fig 6C).

2.6.3 Flat views of volumetric data

Any quantity defined for the voxels in the source volume, such as region annotations, injection densities, etc., can be assigned to their flat projections, resulting in a *flat view* (2D representation of 3D data) of such quantity. Because of this decrease in dimensionality, some aggregation function must be applied to assign the value at a flat point using the combined values of the voxels in its pre-image. Once we have the values, we can use a color palette to obtain a heatmap of the quantity of interest.

To the flat view of region annotations we call a *flatmap* as well, and we use the mode (most common value) as the aggregation function. For flat views of other volumetric datasets, we typically use the mean or maximum (similar to a maximum intensity projection) as aggregation function.

2.6.4 Flat views of long-range axons

We downloaded in bulk the latest MouseLight data release from `ml-neuronbrowser.janelia.org` (dated 2022.02.28) and worked with JSON morphologies in the CCFv3 space. We extracted the locations of the soma and all the axon points, and computed their flat projections based of the voxels to which they belong. Then, we plotted the flat projections of soma and axon points on top of the flatmap of mouse cortical regions for both hemispheres.

3 Results

3.1 Enhancement and flatmapping of rat somatosensory cortex

We apply our methods to an atlas of the somatosensory cortex of juvenile rat. This atlas was built in-house starting from the digital component of Paxinos & Watson's printed atlas (Paxinos and Watson, 2007), consisting of vector drawings of coronal slices with annotated region outlines. In short, the slices were processed: aligned, rasterized and interpolated, to obtain an isotropic voxelized atlas with numeric region labels corresponding to the region and acronym index in the book. The somatosensory regions in one hemisphere were then extracted and smoothed to create an atlas of rat somatosensory cortex. This atlas was downscaled by a factor of $2082/2150 = 0.96837$ (based on cortical thickness of the S1HL region) to approximate the size of a P14 rat brain.

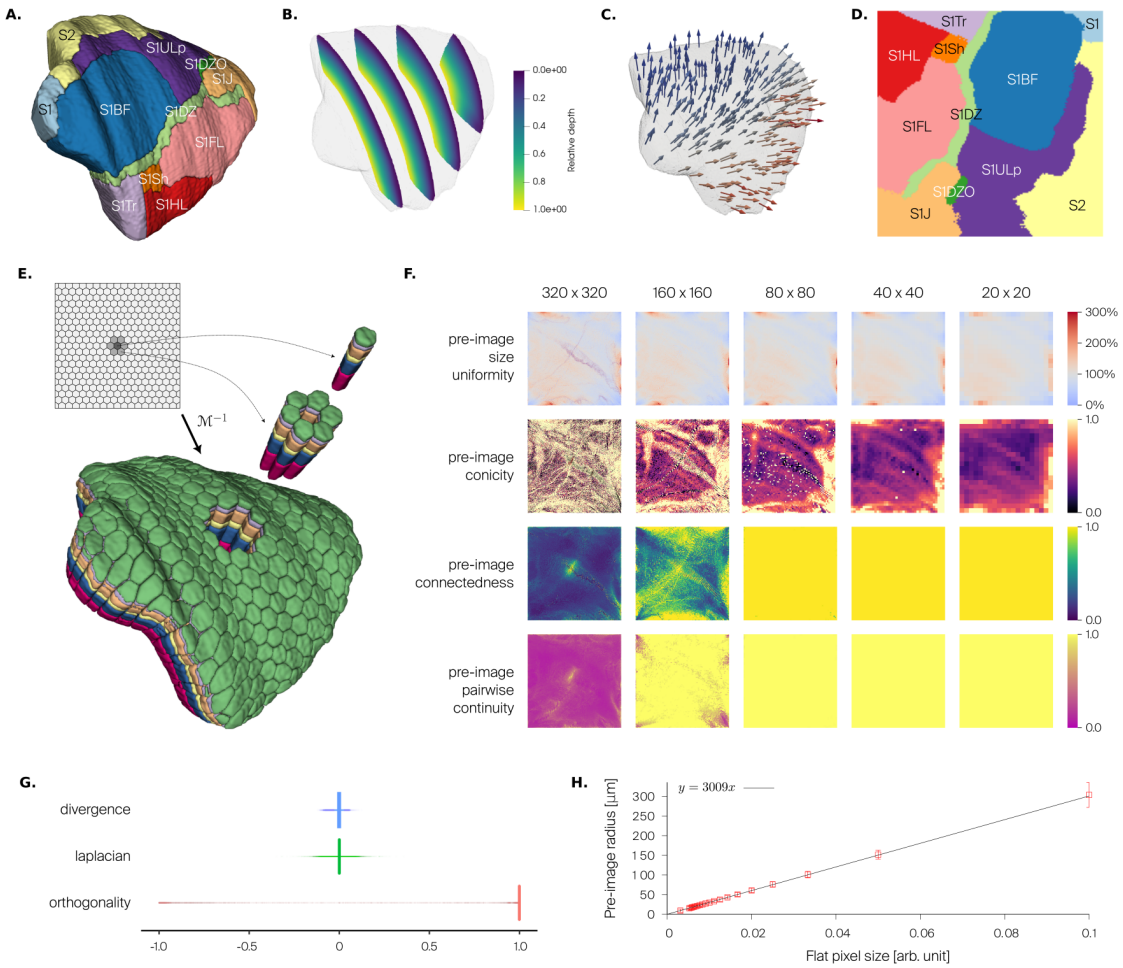

Figure 4: **Enhancement and flatmapping of rat somatosensory cortex. A.** 3D atlas with region annotations. **B.** Relative depth field (sections). **C.** Orientation field. **D.** Flatmap with region annotations. **E.** Decomposition of atlas into hexagonal subvolumes using the inverse flatmap operator $\mathcal{M}^{-1}$. Preservation of layer structure (in colors) is evident in displayed hexagonal columns. Inset shows hexagonal grid drawn on flat space, with highlight of central hexagon (dark gray) and surrounding hexagons (light gray). **F.** Quality metrics as a function of pixel resolution. For pre-image size uniformity, the color bar indicates the proportion of voxels mapped to a pixel compared to that expected from pixel area alone in the perfectly uniform case. We show power-of-two pixel resolutions to have nice sub-sampling in plots. In general, degradation is visible near the edges. **G.** Per-voxel metrics: divergence of orientation field, Laplacian of relative depth field, and orthogonality between principal axis and flatmap axes. **H.** Distance metric obtained from linear fit of flat pixel size vs. pre-image radius (mean and std. dev.).

The final atlas consists of an array of $409 \times 608 \times 286$ voxels of size 38.7348 μm, with 10 annotated regions (Fig. 4A): unlabeled primary somatosensory (S1, light blue), primary somatosensory barrel field (S1BF, blue), primary somatosensory dysgranular zone (S1DZ, light green), primary somatosensory dysgranular oral zone (S1DZO, green), primary somatosensory front limb (S1FL, light red), primary somatosensory hind limb (S1HL, red), primary somatosensory jaw (S1J, light orange), primary somatosensory shoulder (S1Sh, orange), primary somatosensory trunk (S1Tr, light purple), primary somatosensory upper lip (S1ULp, purple), secondary somatosensory (S2, light yellow).

After manual expert selection of top and bottom shells, we obtain the relative depth field (Fig. 4B) and orientation field (Fig. 4C) sampling $N = 1000$ nearest line segments at each voxel.

For flatmapping, the source volume consists of all somatosensory regions, encompassing $1'314,885$ voxels. The projection surface is taken as the isosurface of the relative depth field at $d^* = 0.5$. The reconstructed projection mesh has $23,314$ vertices and an area of 40.606 mm^2, and it was refined three successive times into $1'492,096$ vertices.

The resulting flatmap (Fig. 4D) shows clearly delineated regions. It has a usage fraction at native resolution of 0.662 with a mean pre-image size of 1.5 voxels, and the coverage is 0.988 with only $15,677$ non-eligible voxels.

We show a decomposition of the source volume into a hexagonal column tiling (see Methods; Fig. 4E). We note that the hexagonal shape of the grid defined in flat space is well preserved in Cartesian space. Further, each column preserves the layer structure intact. Potential uses of such a decomposition lie in the analysis of spatially distributed data sets. For example data points associated with neurons can be split up by their soma location into equivalent groups. Statistical testing between the groups can then be used to reveal functional gradients along the horizontal axes. This is especially useful in the context of modeling and simulation, where neurons can be densely sampled. Additionally, individual subvolumes can serve as the target for specific *in silico* experiments.

We investigated the impact of pixel resolution on the properties of pre-images by computing a number of metrics (see Methods 2.5; Fig. 4F). We show resolutions from 320×320 pixels, which is closer to native resolution and is better for drawing on flat space, to 20×20 pixels, which is close to the resolution at which we decompose the source volume, e.g., into hexagonal columns as above. We found the resulting pre-image sizes to be relatively uniform at all resolutions, indicating that different parts of the flat space are comparable in how they sample the Cartesian space. However, at a resolution of 320×320 pixels, the remaining variability still resulted in a number of empty pre-images, i.e. holes. Pre-image conicity was more noisy at high resolutions, probably resulting from the uneven surface structure of the source volume (Fig. 4A vs. 4F, third row). At resolutions below 80×80 pixels, this property was smoother and more uniform. Pre-image connectedness and continuity were largely ensured at resolutions of 80×80 or lower, but fell apart for higher resolutions. For all metrics, the locations of largest non-uniformity were at the edges of the geometry.

Per-voxel metrics show good results (Fig. 4G). Computing the Laplacian of the relative depth field as a measure of smoothness shows good results with a very narrow distribution symmetrical around zero and median value of 0.0003 (0.003 inter-quartile range (IQR)). This value is small compared to the range $[0, 1]$ of the relative depth field. Computing the divergence of the orientation field as a measure of smoothness also shows good results with a slightly wide distribution symmetrical around zero and median value of 0.0 (0.013 IQR). We also compute orthogonality of the local orientation vector to the local flatmap axes, obtaining good results with a very sharp distribution near one and median value of 0.999 (0.001 IQR). This demonstrates that the decomposition into one principal axis and two axes orthogonal to it was successful.

Finally, we obtained a linear fit of flat pixel size vs. pre-image radius (Fig. 4H), leading to a distance metric in flat space of 3009 ± 6 μm per flat coordinate unit.

3.2 Flatmapping of mouse isocortex

Owing to the extensibility of our flatmapping method, we use already-existing datasets from the Allen Institute Mouse Brain Atlas to generate a flatmap of the entire mouse isocortex. We take the solution to Laplace equation available in the CCFv3 2017 release at 10 um resolution, and use it as relative depth field (Fig. 5B). Then, we compute numerically its gradient and use it as orientation field (Fig. 5C). Finally, we apply the same algorithm as before, except we use an iterative version of the authalic mapping algorithm (Saboret et al., 2023).

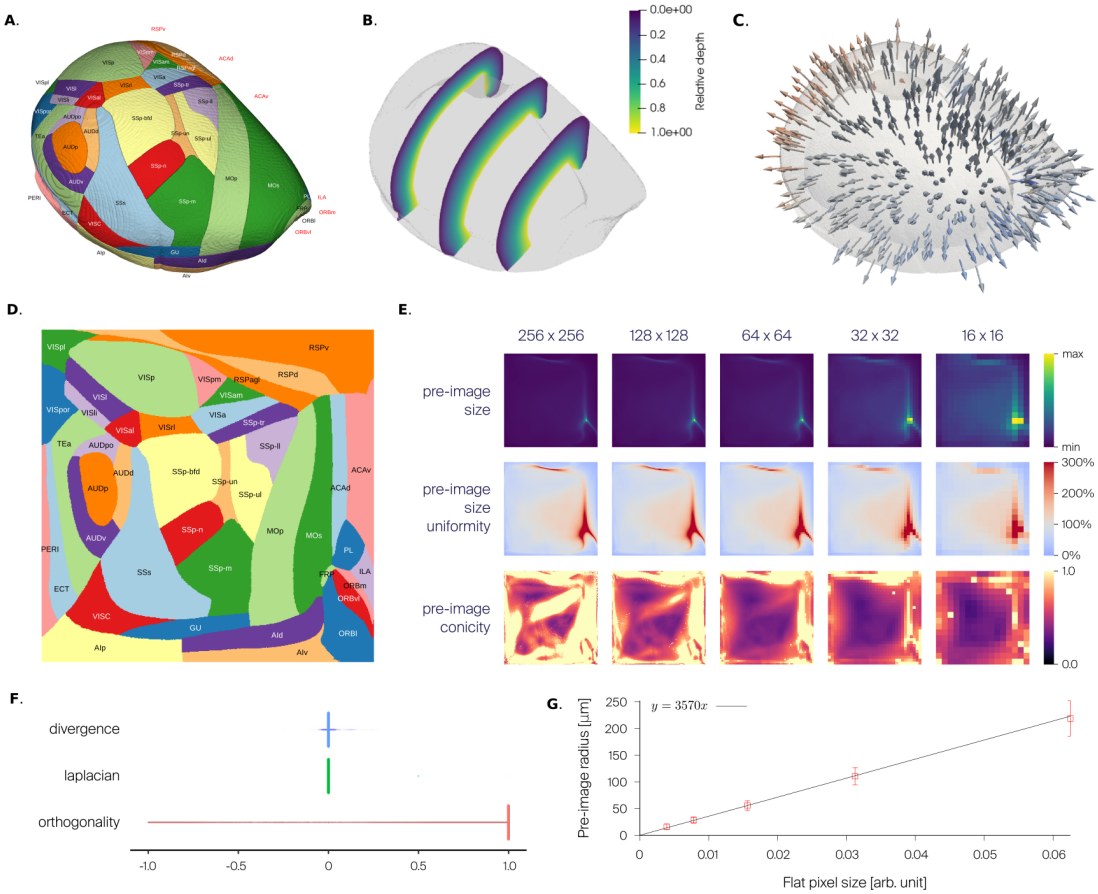

Figure 5: **Flatmapping of mouse isocortex. A.** 3D atlas with region annotations. **B.** Solution to Laplace equation as depth field (sections). **C.** Gradient of the solution to Laplace equation as orientation field. **D.** Flatmap with region annotations. **E.** Quality metrics as a function of pixel resolution. For pre-image size, the color bar indicates the range of values present at each pixel resolution, and cannot be quantitatively compared across resolutions. For pre-image size uniformity, the color bar indicates the proportion of voxels mapped to a pixel compared to that expected from the pixel area alone in the perfectly uniform case. We show power-of-two resolutions to have nice sub-sampling in plots. "Hotspots" in pre-image size are visible in the frontal pole and other regions of high cortical curvature. **F.** Per-voxel metrics: divergence of orientation field, Laplacian of relative depth field, and orthogonality between the principal axis and flatmap axes. **G.** Distance metric obtained from a linear fit of flat pixel size vs. pre-image radius.

We took all 43 regions under "Isocortex" in the hierarchy of the CCFv3 as the source volume (Fig. 5A), encompassing $61'945,750$ voxels. The availability of top, bottom and side shells in the CCFv3 release would allow us to derive the relative depth and orientation fields with our methods. However, we prefer to use the already available solution to the Laplace equation to showcase integration with external datasets derived using different methods from ours. When computing the gradient, we extend the solution one voxel towards the outside to reduce edge artifacts in the numerical derivatives. We take as projection surface the isosurface of the Laplacian solution at $d^* = 0.5$. The reconstructed projection mesh has $437,186$ vertices and an area of 51.652 mm^2, and it was refined three successive times into $27'942,965$ vertices.

The resulting flatmap shows once more clearly delineated regions (Fig. 5D). It has a usage fraction at native resolution of 0.377 with a mean pre-image size of 2.7 voxels, and the coverage is 0.997 with only $213,025$ non-eligible voxels.

In terms of pre-image properties (Fig. 5E), we found that the resulting flatmap was almost completely connected and continuous without holes at resolutions up to 256×256 pixels (not shown for simplicity). Higher pixel resolutions could not be investigated due to computational costs stemming from the high resolution ($10\mu m$) of the input atlas, but based on the behavior of the metrics for the rat flatmap above, a breakdown of connectedness would be expected at a resolution of 1024×1024 pixels, and the appearance of holes would be expected at a resolution of 2048×2048 pixels. Pre-image conicity was smoother than for rat at high resolutions, reflecting the smoothness of the input atlas. One noteworthy feature visible in the pre-image size and uniformity at all pixel resolutions is the presence of "hotspots" in the frontal pole region and in regions along the edge where the cortical hemispheres meet. These are places of high curvature in the cortex, where due to convergence of orientation vectors many voxels are mapped to the same pixel.

The per-voxel metrics showed excellent results (Fig. 5F). As expected, the Laplacian of the depth field (itself a solution to Laplace equation) was very close to zero, with a median of 1.0×10^{-5} (2×10^{-5} IQR), and the divergence of the orientation field was correspondingly also very close to zero, with a median of 1.1×10^{-5} (2×10^{-5} IQR). Orthogonality showed once more a successful decomposition into a principal axis and two orthogonal axes, with a median of 0.999 (0.0013 IQR).

Finally, we obtained a distance metric in flat space with 3570 ± 52 μm per flat coordinate unit (Fig. 5G).

Since the CCFv3 atlas is perfectly symmetrical with respect to the midline, we extended our flatmap to both hemispheres by mirroring, adjusting the flat coordinates to have x ranging in $[0, 1]$ for the right hemisphere and $[1, 2]$ for the left hemisphere. We use this two-hemisphere flatmap for data visualization (Fig. 7C, E).

3.3 Enhancement of mouse barrel cortex

We produced annotations of the posterior medial barrel column field in the global coordinate space of the mouse brain atlas, at 25 um resolution. This is based on flatmapping barrel annotations (Fig. 6A) and considering their pre-image (see Methods). The final barrel annotations encapsulate complete segments of layer 4 creating cylinder-shaped barrels (Fig. 6B).

Our efforts allowed us to derive barrel and barrel column annotations with the following properties. Where neighboring barrels are separated by the septum, the associated barrel columns will also be separated throughout the entire depth; where the barrels touch, the columns will touch, but not overlap throughout the entire depth. The shape of the columns follows the curvature of the cortex. (Fig. 6C).

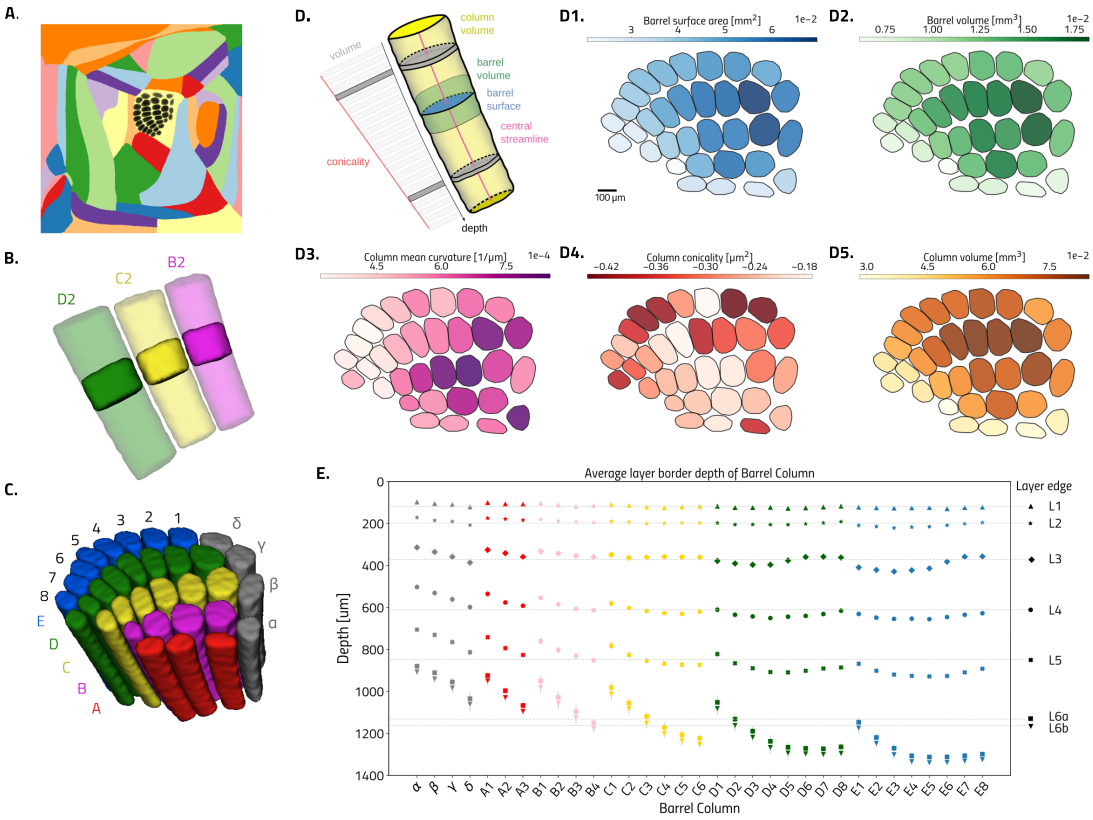

Figure 6: **Enhancement of mouse barrel cortex.** **A.** Flat projections of annotated barrel voxels (see Methods, black) in SSp-bfd, shown in the context of the full mouse isocortex. **B.** Barrel volumes of 2-row (solid) and their corresponding columns, defined as the pre-images of their flat projections (semi-transparent). **C.** Extracted barrel columns for all annotated barrels, barrel rows represented by the same color. **D.** Schematic of quantification metrics. Followed by the intensity map representation of following features : **D1.** Barrel surface area. Shown as a cross-section (blue) in the schematic. **D2.** Barrel volume. Shown as an outline (green) in the schematic. **D3.** Column mean curvature. Computed as the mean curvature of the central streamline (magenta) shown in the schematic. **D4.** Column conicality. Computed as the slope of a linear fit (red) to a depth-wise volume histogram (gray) shown in the schematic. **D5.** Column volume. Shown as a volume (yellow) in the schematic. **E.** Average layer border depth for all layers in each of the barrel columns. Grey lines indicated average layer depths across the barrel columns.

To further characterize the geometry of the new annotations, we computed a number of anatomical metrics (Fig. 6D). For the surface of each barrel we considered the convex hull of the flat projections of the barrel voxels and calculated their area as previously described (Fig. 6 D1, see Methods 2.3). The resulting surfaces have an elongated shape, in some cases twice as wide along an arc than along a row, in line with (Welker and Woolsey, 1974, Fig. 6 D1). Furthermore, barrels within a row are very close to each other, often touch, but do not overlap, whilst big septal areas can be observed between the rows.

We then calculated the volumes of barrels and barrel columns, based on the voxel count and voxel size (Fig. 6 D2, D5). In terms of volume and surface area barrels were generally smaller towards the periphery (Fig. 6 D1, D2), with barrel D1 being the largest. Column volume was strongly, but not completely dependent on barrel surface area. An additional factor was the increase of cortical thickness from row A to row E (Fig. 6 D5, E).

Additionally, we characterized how much the columns deviate from right cylinders, measuring their *conicality* and *curvature* (Fig. 6 D3, D4). The first metric is the slope of a linear fit to the depth-wise volume histogram. The second metric is defined as the mean curvature of the central streamline, calculated with Frenet-Serret formulas (Frenet–Serret, 2023). Over the entire depth, the C2 barrel curves 160 μm towards the C1 barrel column compared to a corresponding representation as a right cylinder. As a result of the global curvature of the cortical volume, columns were wider at the top than at the bottom. The strength of this effect was independent of column curvature (Fig. 6 D4). The approximated radius of the C2 barrel column narrowed by 32 μm over the entire depth.

Finally, we found the layer borders and quantified them depth-wise along the barrel columns (Fig. 6E). We found that the barrel columns increase in height from row A to E. We can also observe a pattern of height increasing with the number of the arc. However, the relative layer depths are largely preserved across the barrel columns.

3.4 Applications of flatmaps

Many applications of the flatmap lie in the field of large-scale, bio-physically detailed modeling. For example, one can use the depth field to define a layer segmentation in an atlas that lacks one; one can use the orientation field to correctly orient reconstructed neuron morphologies during the building of a detailed model; finally, the depth field can also be used to select reconstructed morphologies that can be placed at a given location without protruding beyond layer boundaries. These use cases as applied to rat somatosensory cortex are explained in (Reimann et al., 2022), so here we will demonstrate additional applications, many of them in mouse.

3.4.1 Topographical mapping for long-range connectivity

A flat view of cortical regions can be used to depict and parameterize the topographical mapping of connections between them. This principle has been used to predict a neuronal resolution long-range connectome in (Reimann et al., 2019). In short, the spatial targeting of cortico-cortical connections is often expressed in terms of laminar synapse profiles, i.e. its specificity along the principal axis orthogonal to layer boundaries. However, axons from individual neurons are also targeting specific locations along the other two dimensions, and this can be expressed in terms of a topographical mapping between points in flat space. We parameterized this in the form of pairs of triangles defining local barycentric coordinates, with the assumption that neurons at a given location of one triangle predominately project to the corresponding point of the other triangle (Fig. 7B).

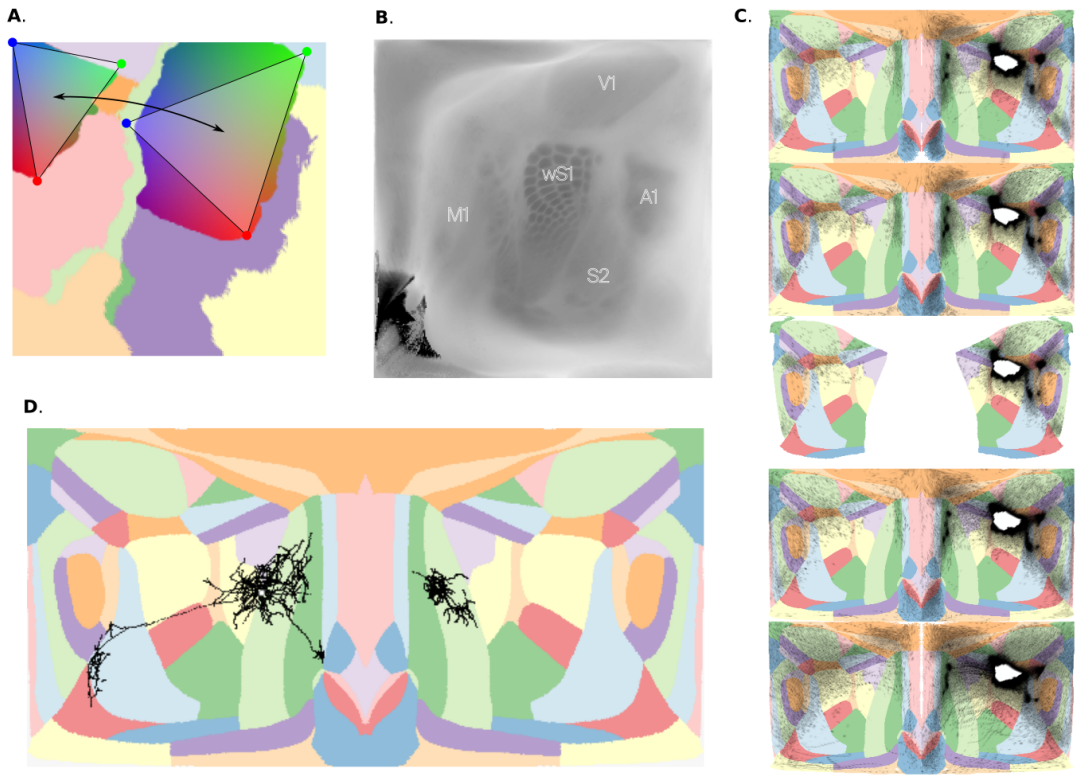

Figure 7: **Applications of flatmaps. A.** Topographical mapping between two rat somatosensory regions. Locations with the same color, derived from barycentric coordinates of triangles contained in each region (red, green and blue vertices), connect to each other. **B.** Flat view of the CCFv3 average template of mouse isocortex. **C.** Flat views of injection (white) and projection density (black) across cortical layers in both hemispheres of mouse isocortex. Per-layer maps are stacked, and correspond to L1, L2/3, L4, L5 and L6. **D.** Flat view of long-range axons (black points), for a neuron with soma (white dot) at the edge of lower limb primary somatosensory cortex (SSp-ll) and primary motor cortex (MOp).

3.4.2 Volumetric data visualization

The flatmap allows us to visualize volumetric data in 2D, as long as the data is registered to the same brain atlas that was flatmapped. We show flat views of the "average template" dataset part of CCFv3 (Fig. 7B, Wang et al., 2020) and of projection data from the Allen Mouse Connectivity Atlas (Fig. 7C, Oh et al., 2014), using our flatmap of mouse isocortex.

The average template (Fig. 7B) is constructed by interpolation of tissue auto-fluorescence data from high-resolution serial two-photon tomography images of 1,675 young adult mouse brains. The flat view was obtained using the maximum as aggregation function, resulting in a flat maximum intensity projection (fMIP). Indeed, the resulting image is very similar to a cytochrome oxidase staining of mechanically flattened cortex (compare Fig. 7B to Fig. 1 of Airey et al., 2005).

For the connectivity data, we additionally split the flat view into layers to show information with laminar specificity (Fig. 7C). Specifically, we show the injection and projection densities of connections originating mainly from the SSp-bfd region (some injection density in VISrl) in a single experiment (ID 126907302). From the image we can see the strongest cortical targets were parts of the ipsilateral secondary somatosensory and motor areas, the AUDd auditory area, and the neighboring VISal and VISa visual areas. Weaker targets, which suffer from high noise, were the remaining visual and auditory regions, and some retrosplenial, temporal and orbital areas. Other prefrontal and anteromedial areas, including most of primary somatosensory cortex, were largely avoided.

3.4.3 Long-range axon visualization

We used the flatmap of mouse isocortex to visualize the anatomy and targeting of long-range axons, using data from Janelia MouseLight (mouselight.janelia.org; Economo et al., 2016). In the absence of a flatmap, such visualizations typically depict axonal morphology in the form of a parallel projection of a slice. In contrast, our approach supports viewing axon collaterals that do not lie in the same plane simultaneously. Furthermore, it visualizes the horizontal targeting of axons independently from the vertical one, i.e. independently from their laminar termination profiles. We show an exemplary axon (ID AA0002, Fig. 7D) originating from the primary motor cortex (MOp) near the border with primary somatosensory lower limb region (SSp-ll). We can observe branches innervating nearby parts of SSp-ll, as well as ipsilateral secondary somatosensory and motor regions with high specificity. Other branches innervate motor regions in the contralateral hemisphere. As the flatmap covers only the isocortex, parts of the axon leaving this region are not visible. Therefore the contralateral innervating parts appear to be separated from the rest and branches innervating subcortical structures are missed completely.

4 Discussion

We have demonstrated a method for the generation of coordinate systems adapted to the geometry of laminar regions in digital brain atlases, along with a number of applications highlighting their utility. While this is not the first time the concept of a flatmap has been introduced, we described a general method that is applicable to any region with a primary axis of organization and have made our implementation publicly available (see Code and data availability). Additionally, we moved beyond the purely flat view and also offer a full description of the third, orthogonal axis, including its local orientation.

Furthermore, we have defined a set of metrics that can be used to evaluate a flatmap in terms of its overall quality and suitability for different applications. The metrics can also be used to find an appropriate discrete resolution, depending on the intended use. For visualization purposes, the primary concern would be to avoid the presence of empty pre-images, i.e. holes in the flat view. Additionally, use cases such as long-range axon visualization require spatial continuity of the flat view, which may require a lower resolution. This is especially crucial when data is first flatmapped and then gradients along the flat coordinates are considered (Guyonnet-Hencke and Reimann, 2023), since spatial continuity ensures that a gradient in flat space corresponds to a gradient along a continuous line in Cartesian coordinates. Conversely, volume decomposition (and similarly, definition of barrels) does not require this and can even be conducted at native resolution. Further, while our algorithms are generally robust, the results are less noisy when the input atlas is smooth; the uniformity of pre-image sizes can be helpful in finding a resolution that evens out the noise.

One additional parameter that can be optimized to the application is the selection of the depth d^* at which the projection surface is taken. Due to the nature of the algorithm used to flatten the mesh, preservation of relative areas and continuity of the flat view is guaranteed at the selected depth. But as one moves further away from that depth, these properties can degrade, depending on the quality of the orientation field and consequently, the streamlines. In our examples, we placed the projection mesh in the exact middle of the volume, but use cases that focus on more superficial or deeper layers may benefit from a different placement of the projection surface.

Taken together, we conclude that a single coordinate system cannot be ideal for all use cases. This demonstrates the importance of an open, standardized algorithm whose parameters can be optimized by the user, rather than a static, fixed solution.

We found our methods to be sensitive to edge artifacts. Locations with the largest non-uniformity and highest probability of non-continuity were usually in the periphery of the flat space. A possible cause for this may be the forced stretching of the projection mesh into a square, i.e. its outside border being made into straight lines regardless of its original shape. While an alternative parameterization in terms of a circular border is readily available, this would be problematic for discretization into square pixels as is required for visualization. A mitigation strategy would be then to extend the projection mesh to a square before flattening, so that the border would map naturally to the flat unit square. This would also result in a depiction of the regions of interest with flat shapes more akin to their 3D shapes. Additionally, a relatively small number of voxels at the edges of the source volume were non-eligible. Where possible, this can be mitigated by taking a larger source volume, effectively displacing the edges to the surrounding regions and leaving the actual regions of interest in the interior. Another option is to adopt a weaker eligibility criterion and keep voxels as as long as their streamlines reach the projection surface, even if they do not span the whole depth. In any case, having a smoother input atlas, either from better data or higher resolution, can usually help in this case. While we believe these issues may be of secondary importance for the rodent cortex, they may may be worthwhile for regions with more complex geometries such as hippocampus or cerebellum.

When flattening mouse isocortex, the pre-image metrics consistently depicted a "hotspot" at the frontal pole areas. This is caused by the anatomy of those areas featuring an extreme narrowing towards layer 6. Specifically, in slices of those areas, the top of layer 1 can be more than four times longer than the bottom of layer 6 (Van De Werd et al., 2010). As such, this does not indicate a failure of our algorithm, but an intrinsic limitation of the streamline approach as applied to the geometry of the cortex. In the future, it may be instructive to consider the anatomical and functional implications of this.

Making use of the atlas enhancements, we successfully increased the granularity of the CCFv3 annotations in barrel cortex, by defining 33 barrels and their associated columns. Segmentation of additional barrels was unsuccessful due to the limited signal-to-noise ratio of the average template dataset. The resulting barrel annotations have the following properties. Firstly, they match previous accounts of barrels (Welker and Woolsey, 1974; Lefort et al., 2009) in terms of shapes and sizes. Secondly, they are defined in the standardized CCFv3 space, which enables the integration and analysis of datasets with single-barrel specificity. Finally, the barrel columns are non-overlapping and follow the curvature of the cortex, resulting in a more biologically accurate representation.

These new annotations provide a reference framework for future studies in barrel cortex, such as anatomical quantification, connectivity tracing (cortico-cortical and thalamocortical), and computational modeling.

Code and data availability

Enhanced atlases and flatmaps, as well as example code to work with them, are made available in a public repository under a CC BY-NC-SA license: https://zenodo.org/record/8165005.

Code implementing the flatmapping algorithm will be made available in a public repository under a free software license.

References

Airey, D. C., Robbins, A. I., Enzinger, K. M., Wu, F., and Collins, C. E. (2005). Variation in the cortical area map of C57BL/6J and DBA/2J inbred mice predicts strain identity. *BMC Neuroscience*, 6(1):18.

DeKraker, J., Ferko, K. M., Lau, J. C., Köhler, S., and Khan, A. R. (2018). Unfolding the hippocampus: An intrinsic coordinate system for subfield segmentations and quantitative mapping. *NeuroImage*, 167:408–418.

Economo, M. N., Clack, N. G., Lavis, L. D., Gerfen, C. R., Svoboda, K., Myers, E. W., and Chandrashekar, J. (2016). A platform for brain-wide imaging and reconstruction of individual neurons. *eLife*, 5:e10566.

Egger, R., Narayanan, R. T., Helmstaedter, M., Kock, C. P. J. d., and Oberlaender, M. (2012). 3D Reconstruction and Standardization of the Rat Vibrissal Cortex for Precise Registration of Single Neuron Morphology. *PLOS Computational Biology*, 8(12):e1002837. Publisher: Public Library of Science.

Frenet–Serret (2023). Frenet–Serret formulas. Page Version ID: 1166216360.

Galassi, M., editor (2009). *GNU scientific library reference manual: for GSL version 1.12*. Network Theory, Bristol, 3. ed edition.

Geuzaine, C. and Remacle, J.-F. (2009). Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331.

Guyonnet-Hencke, T. and Reimann, M. W. (2023). A parcellation scheme of mouse isocortex based on reversals in connectivity gradients. *Network Neuroscience*, pages 1–23.

Jones, S. E., Buchbinder, B. R., and Aharon, I. (2000). Three-dimensional mapping of cortical thickness using Laplace's Equation. *Human Brain Mapping*, 11(1):12–32.

Lefort, S., Tomm, C., Floyd Sarria, J.-C., and Petersen, C. C. (2009). The Excitatory Neuronal Network of the C2 Barrel Column in Mouse Primary Somatosensory Cortex. *Neuron*, 61(2):301–316.

Lerch, J. P., Carroll, J. B., Dorr, A., Spring, S., Evans, A. C., Hayden, M. R., Sled, J. G., and Henkelman, R. M. (2008). Cortical thickness measured from MRI in the YAC128 mouse model of Huntington's disease. *NeuroImage*, 41(2):243–251.

Oh, S. W., Harris, J. A., Ng, L., Winslow, B., Cain, N., Mihalas, S., Wang, Q., Lau, C., Kuan, L., Henry, A. M., Mortrud, M. T., Ouellette, B., Nguyen, T. N., Sorensen, S. A., Slaughterbeck, C. R., Wakeman, W., Li, Y., Feng, D., Ho, A., Nicholas, E., Hirokawa, K. E., Bohn, P., Joines, K. M., Peng, H., Hawrylycz, M. J., Phillips, J. W., Hohmann, J. G., Wohnoutka, P., Gerfen, C. R., Koch, C., Bernard, A., Dang, C., Jones, A. R., and Zeng, H. (2014). A mesoscale connectome of the mouse brain. *Nature*, 508(7495):207–214.

Papp, E. A., Leergaard, T. B., Calabrese, E., Johnson, G. A., and Bjaalie, J. G. (2014). Waxholm Space atlas of the Sprague Dawley rat brain. *NeuroImage*, 97:374–386.

Paxinos, G. and Franklin, K. (2019). *Paxinos and Franklin's the Mouse Brain in Stereotaxic Coordinates*. Elsevier, Academic Press, London San Diego Cambridge; MA Kidlington, Oxford, 5. edition edition.

Paxinos, G. and Watson, C. (2007). *The rat brain in stereotaxic coordinates*. Elsevier, Amsterdam, 6th ed edition.

Povolotsky, A. V., Gevaert, M., Riquelme, L., Sanin, A., Courcol, J.-D., Guyot, L., Ficarelli, G., Vanherpe, L., Dictus, H., Zisis, E., Dietz, A., Berchet, A., O'Reilly, C., Šubert, E., and Antonel, S. (2023). BlueBrain/voxcell: v3.1.5.

Reimann, M. W., Bolaños-Puchet, S., Courcol, J.-D., Santander, D. E., Arnaudon, A., Coste, B., Delemontex, T., Devresse, A., Dictus, H., Dietz, A., Ecker, A., Favreau, C., Ficarelli, G., Gevaert, M., Hernando, J. B., Herttuainen, J., Isbister, J. B., Kanari, L., Keller, D., King, J., Kumbhar, P., Lapere, S., Lazovskis, J., Lu, H., Ninin, N., Pereira, F., Planas, J., Pokorny, C., Riquelme, J. L., Romani, A., Shi, Y., Smith, J. P., Sood, V., Srivastava, M., Van Geit, W., Vanherpe, L., Wolf, M., Levi, R., Hess, K., Schürmann, F., Muller, E. B., Ramaswamy, S., and Markram, H. (2022). Modeling and Simulation of Rat Non-Barrel Somatosensory Cortex. Part I: Modeling Anatomy. *bioRxiv*.

Reimann, M. W., Gevaert, M., Shi, Y., Lu, H., Markram, H., and Muller, E. (2019). A null model of the mouse whole-neocortex micro-connectome. *Nature Communications*, 10(1):3903.

Romani, A., Antonietti, A., Bella, D., Budd, J., Giacalone, E., Kurban, K., Sáray, S., Abdellah, M., Arnaudon, A., Boci, E., Colangelo, C., Courcol, J.-D., Delemontex, T., Ecker, A., Falck, J., Favreau, C., Gevaert, M., Hernando, J. B., Herttuainen, J., Ivaska, G., Kanari, L., Kaufmann, A.-K., King, J. G., Kumbhar, P., Lange, S., Lu, H., Lupascu, C. A., Migliore, R., Petitjean, F., Planas, J., Rai, P., Ramaswamy, S., Reimann, M. W., Riquelme, J. L., Guerrero, N. R., Shi, Y., Sood, V., Sy, M. F., Geit, W. V., Vanherpe, L., Freund, T. F., Mercer, A., Muller, E., Schürmann, F., Thomson, A. M., Migliore, M., Káli, S., and Markram, H. (2023). Community-based Reconstruction and Simulation of a Full-scale Model of Region CA1 of Rat Hippocampus.

Saboret, L., Alliez, P., Lévy, B., Rouxel-Labbé, M., Fabri, A., and Jain, H. (2023). Triangulated surface mesh parameterization. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.2 edition.

Simons, D. J. and Woolsey, T. A. (1979). Functional organization in mouse barrel cortex. *Brain Research*, 165(2):327–332.

Sumser, A., Mease, R. A., Sakmann, B., and Groh, A. (2017). Organization and somatotopy of corticothalamic projections from L5B in mouse barrel cortex. *Proceedings of the National Academy of Sciences*, 114(33):8853–8858.

Tange, O. (2011). GNU Parallel - The Command-Line Power Tool. *login: The USENIX Magazine*, pages 42–47.

Tangelder, H. and Fabri, A. (2023). dD spatial searching. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.2 edition.

Van De Werd, H. J. J. M., Rajkowska, G., Evers, P., and Uylings, H. B. M. (2010). Cytoarchitectonic and chemoarchitectonic characterization of the prefrontal cortical areas in the mouse. *Brain Structure and Function*, 214(4):339–353.

van Lankveld, T. (2023). Scale-space surface reconstruction. In *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.2 edition.

Wang, Q., Ding, S.-L., Li, Y., Royall, J., Feng, D., Lesnar, P., Graddis, N., Naeemi, M., Facer, B., Ho, A., Dolbeare, T., Blanchard, B., Dee, N., Wakeman, W., Hirokawa, K. E., Szafer, A., Sunkin, S. M., Oh, S. W., Bernard, A., Phillips, J. W., Hawrylycz, M., Koch, C., Zeng, H., Harris, J. A., and Ng, L. (2020). The Allen Mouse Brain Common Coordinate Framework: A 3D Reference Atlas. *Cell*, 181(4):936–953.e20.

Welker, C. and Woolsey, T. A. (1974). Structure of layer IV in the somatosensory neocortex of the rat: Description and comparison with the mouse. *The Journal of Comparative Neurology*, 158(4):437–453.

Yushkevich, P. A., Gao, Y., and Gerig, G. (2016). ITK-SNAP: An interactive tool for semi-automatic segmentation of multi-modality biomedical images. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3342–3345, Orlando, FL, USA. IEEE.

Supplementary material

Implementation details

1. Extraction of the point cloud from which the projection mesh is reconstructed is implemented in Python using the Euclidean Distance Transform algorithm in SciPy.
2. Reconstruction of the projection mesh is implemented in C++ using the Scale-Space Surface Reconstruction package from CGAL (van Lankveld, 2023).
3. Integration of streamlines is implemented in C using a numerical ODE integrator from GSL (Galassi, 2009).
4. Parametrization of streamlines is implemented in C using a linear spline interpolator from GSL.
5. Projection of voxel centers to projection surface is implemented in C using a root finding algorithm from GSL.
6. Refinement of the projection mesh by uniform subdivision is performed with GMSH (Geuzaine and Remacle, 2009).
7. Mapping of streamline intersections to mesh points is implemented in C++ using Nearest Neighbor search from the dD Spatial Search CGAL package (Tangelder and Fabri, 2023).
8. Flattening of the projection mesh is performed using the (Iterative) Authalic Mapping algorithm from the Triangulated Surface Mesh Parameterization CGAL package (Saboret et al., 2023).
9. Discretization of the flatmap is performed in Python using NumPy.
10. Handling of atlas volumes in NRRD format is performed in Python using voxcell (Povolotsky et al., 2023).
11. The workflow orchestrating all steps in the algorithm is written as a GNU Makefile.
12. Computation of streamlines in parallel is performed with GNU Parallel (Tange, 2011).
13. Plotting of flat views is performed in Python using Datashader.
14. All code was run under GNU/Linux on the BB5 supercomputer hosted at the CSCS Swiss National Supercomputing Centre.

Flatmapping of non-layered brain structures

Our flatmapping method can also be applied to non-layered brain regions, as long as a principal axis can be defined that is meaningful for the geometry of the structure under study.

Here we provide as an example the generation of a flatmap for the dorsal striatum of mouse. This structure has no cytoarchitectonically defined layers, but it has a smooth organic shape with a meniscus cross-section (similar to a cupped hand) that can be approximated by an ellipsoid. The principal axis is then defined as the normal vector to the surface of the ellipsoid, and the flatmap axes go along the surface of the ellipsoid.

In detail:

1. We extract the voxels belonging to the dorsal striatum (STRd) region from the Allen Mouse Brain Atlas CCFv3 at 25 um resolution.
2. We fit a plane to the point cloud defined by the voxel centers, in order to get an idea of the "facing" direction of the structure.

3. We fit a sphere to the point cloud on the concave side, with center along the line defined by the plane normal.

4. Using the position and radius of the sphere as initial condition, we fit an ellipsoid using the sum of orthogonal distances as the cost function with the COBYLA optimization algorithm (NLopt library). The resulting ellipsoid passes through the "middle" of the point cloud, similar to the choice of isosurface with relative depth 0.5 in the main text.

5. We compute the bounding box of the point cloud and intersect this with the ellipsoid to obtain a patch of ellipsoidal surface that we use as projection mesh.

6. We compute the projections of voxel centers as orthogonal projection to the ellipsoid, that is, streamlines are assumed to be straight line segments having the orientation of the surface normals.

7. We compute relative depth by considering the normalized distance to the ellipsoid surface, taking into account which side of the surface the point is in, so we obtain a continuous value in [0,1].

8. The rest of the algorithm proceeds as usual, with authalic flattening of the projection mesh and finding nearest neighbors between intersection points and mesh vertices.

The resulting flatmap provides coordinates along the "face" of the dorsal striatum, and allows a decomposition of the volume of this structure into uniform subvolumes. A similar approach can be applied to all brain structures with a similar shape, e.g. reticular nucleus of thalamus.

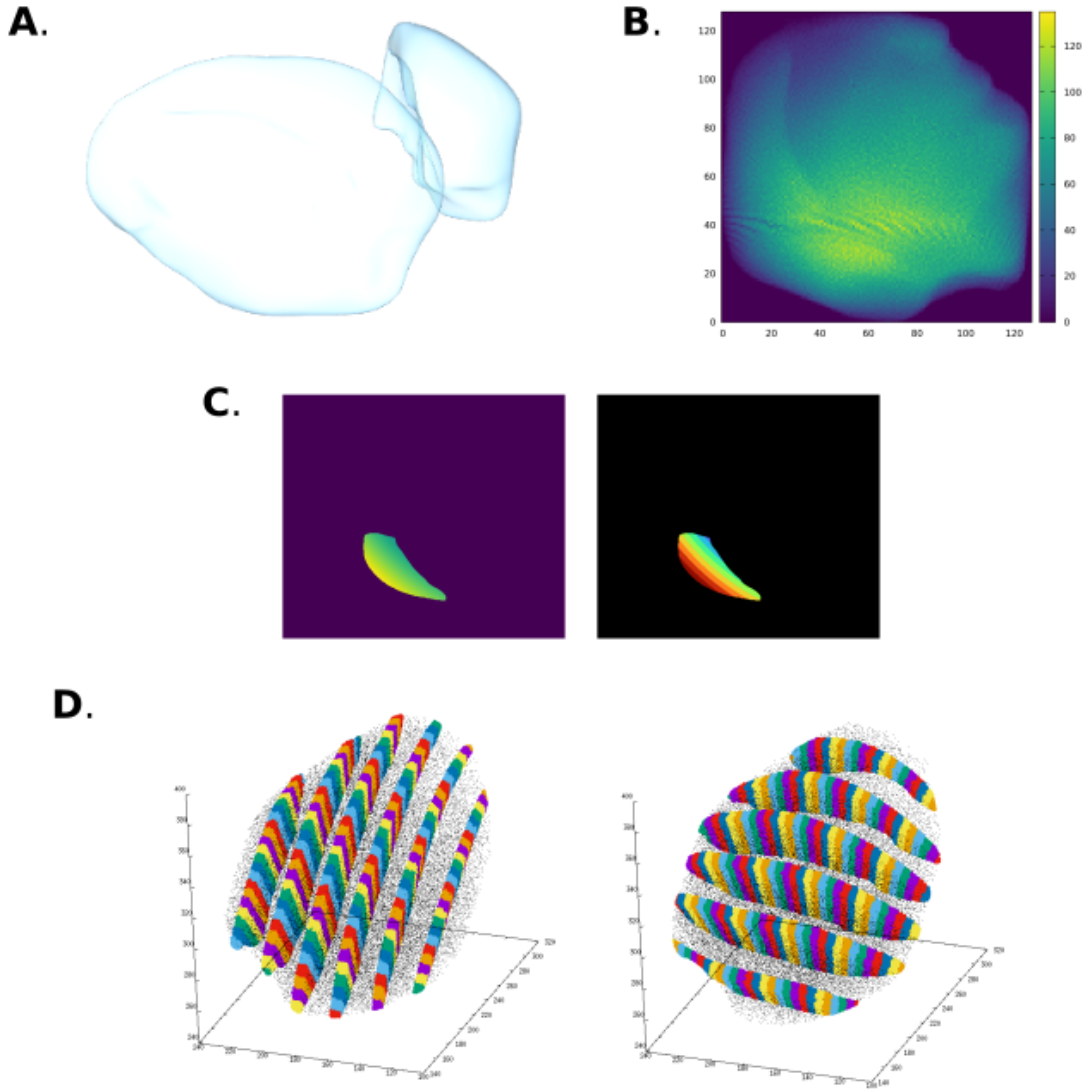

Figure 8: **Flatmapping non-layered brain structures: dorsal striatum. A.** 3D meshes of bilateral dorsal striatum, showing "face" of structure and lateral aspect. **B.** Pre-image size heatmap of resulting flatmap. Note how the edges of the flatmap do not coincide with the edges of the square due to how the projection mesh was chosen. **C.** Slice through the relative depth field, showing continuous variation (left) and discrete levels to exhibit ellipsoidal shape of isosurfaces (right). **D.** Decomposition of the volume of dorsal striatium into square columns. Note how the columns curve following the normals of the ellipsoidal surface and extend in depth.