Computers & Graphics 98 (2021) 138–149

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Special Section on VCBM 2020

Interactive analysis for large volume data from fluorescence microscopy at cellular precision

Yong Wan*, Holly A. Holman, Charles Hansen

The University of Utah, Salt Lake City, 84112, USA

ARTICLE INFO

Article history:
Received 7 January 2021
Revised 13 May 2021
Accepted 14 May 2021
Available online 24 May 2021

Keywords:
Interactive analysis
Volume analysis
Rapid analysis
Cell analysis
Brushing
Fluorescence microscopy
Large data
Streamed processing

ABSTRACT

The main objective for understanding fluorescence microscopy data is to investigate and evaluate the fluorescent signal intensity distributions as well as their spatial relationships across multiple channels. The quantitative analysis of 3D fluorescence microscopy data needs interactive tools for researchers to select and focus on relevant biological structures. We developed an interactive tool based on volume visualization techniques and GPU computing for streamlining rapid data analysis. Our main contribution is the implementation of common data quantification functions on streamed volumes, providing interactive analyses on large data without lengthy preprocessing. Data segmentation and quantification are coupled with brushing and executed at an interactive speed. A large volume is partitioned into data bricks, and only user-selected structures are analyzed to constrain the computational load. We designed a framework to assemble a sequence of GPU programs to handle brick borders and stitch analysis results. Our tool was developed in collaboration with domain experts and has been used to identify cell types. We demonstrate a workflow to analyze cells in vestibular epithelia of transgenic mice.

1. Introduction

Fluorescence microscopy employs fluorescent dyes and genetically encoded fluorescent proteins to reveal target biological structures, such as cells and tissues. Three-dimensional (3D) scans are obtained through optical sectioning using setups such as confocal [1] and two-photon [2] microscopy. Multiple channels of volume data are imaged simultaneously via fluorescent tags of different peak emission wavelengths. The main objective for understanding these co-registered volume channels is to investigate and evaluate the fluorescent signal intensity distributions as well as their relationships. On the one hand, an investigator qualitatively examines the data visualizations and draws conclusions about a hypothesis based on whether fluorescent signals in a predefined region are detected; on the other hand, objectivity and repeatability are emphasized through quantitative and statistically tested analyses. We often combine qualitative and quantitative analyses for real-world data. For example, the colocalization analysis often starts with the examination of the superimposed channels in distinct excitation and emission spectra, continues with the computing of the Pearson Correlation Coefficient [3] or Manders Colocalization Coefficient [4], and concludes with a probability of soundness by comparing the results with those from random samples [5]. As noted in previous research, inclusion or exclusion of the background information can significantly influence the quantitative analysis results [6]. To reduce the bias introduced by the irrelevant intensity signals to an experiment, a subjective pruning process has been recommended for the colocalization analysis [6]. Likewise, many other data analysis tasks need an effective tool with a well-balanced and tightly integrated design, enabling both subjective evaluation and objective quantitation to collaborate. According to this design philosophy, the analysis of 3D fluorescence microscopy data needs interactive tools for visualization and data quantitation. More importantly, users of such a tool will be able to apply data quantitation at designated regions instantly and intuitively.

In practice, several technical challenges exist in building such an interactive tool. The major obstacle is a growing data size from a series of high-resolution microscopy techniques [7,8]. We performed several speed benchmarks to compare the performance between volume rendering and common analysis routines. The results demonstrated that data analyses on the GPU usually demanded higher computational power than rendering the same data set (see the supplementary benchmarks). When an existing visualization tool is extended for quantitative analysis, we estimate the data size that can be interactively processed to be limited to several tens of megavoxels. Common practice in biomedical experiments suggests that such small data size do not meet researchers'

* Corresponding author.
E-mail address: wanyong@cs.utah.com (Y. Wan).

https://doi.org/10.1016/j.cag.2021.05.006
0097-8493/

demands because a single full-resolution scan from conventional confocal microscopy easily reaches several hundred megavoxels. However, in an interactive analysis workflow, the amount of data that needs rapid processing is constrained by a limited scope of momentary user attention. In other words, techniques such as streamed rendering are applicable for interactive data analysis, where GPU-based data processing only applies to a portion of an entire data set at a time under user guidance. For such streamed visualization and analysis, we precompute a large volume into partitions, which are also known as bricks for a regular partitioning grid. Unlike the standard volume rendering in a stream, streamed data analysis presents a challenge as more inter-partition information is required when an integral structure can easily locate on brick borders or span multiple bricks. A systematic approach is thus needed to enable data gathering from multiple bricks, so that a standard workflow is repeatable for various analyses. Furthermore, the linking between user interactions and data analysis workflow needs to be seamless for usability. Since we provide brushing as an intuitive user operation for isolating structures and focusing analysis on relevant features, computations for analysis need to keep track of the brushing results in order to constrain data load and achieve rapid processing.

Here, we present the design of an interactive data analysis tool for 3D fluorescence microscopy data. Our main contribution is the implementation of common data quantification functions on streamed volumes, providing interactive analyses on large data without lengthy preprocessing. The innovative features of the tool include:

(1) Enhanced Synthetic Brainbows for cell segmentation. We added two criteria to the Synthetic Brainbows framework to handle uneven intensity distribution and fused cells in fluorescence data.
(2) Diffusion calculation for brushing on large volume data of multiple bricks. We chose to compute the brushing results using the morphological diffusion to achieve high-quality selection borders. The design allows diffusion to propagate across data bricks and keeps track of selected bricks for efficient data processing.
(3) Brick stitching. We used graphs to connect segmentation results and guide the assemblance of data analysis results from multiple bricks.
(4) Demonstration of usability in real-world biomedical research. Specifically, we identified and classified the progenitor-like cells in the mouse semicircular canals.

2. Related work

We drew our ideas to implement a volume data analysis tool from previous work on interactive visualization, segmentation, and quantitative measurements. We improved a series of techniques for efficient GPU implementations and brought them together for a consistent workflow.

2.1. Brushing for interactive selection of volumetric structures

Many volume data analysis tools support user-defined areas of interest (AOIs). An axis-aligned box is the most commonly used AOI type, which is often created by dragging a computer mouse from orthographic views or directly inputting coordinates. There are also tools that allow lassoing or brushing 2D sections for more freedom in selection. But the quality of a resulting selection of 3D structures is often compromised by unsmooth contours. It also becomes difficult to correctly select complex structures just from 2D sections. Using the volume intersection techniques proposed by Martin and Aggarwal [9] and Kutulakos and Seitz (Space Carving) [10], direct 3D selections from projected 2D views became possible. More sophisticated methods employed sketch-based interactions. Yuan et al. presented a method for cutting out 3D volumetric structures based on simple strokes drawn on volume renderings [11]. Chen et al. enabled sketch-based seed placing for interactive region growing in a volume manipulation tool [12]. Owada et al. proposed several sketching user interface tools for region selection [13]. Abeysinghe and Ju used 2D sketches to constrain skeletonization of intensity volumes [14]. Wan et al. designed a brush tool with dual-stroke stamps for simultaneous seeding and growing to segment neural and cellular structures [15,16]. We based our brushing interactions on a dual-stroke brush for its ability to generate smooth selection masks from fluorescence microscopy data, where structures usually do not possess clear borders as signal intensity drops gradually. The difficulty of applying the diffusion-based volume selection for large data is finding the smallest sub-volume that only contains the diffusion flow. Our implementation partitions a large data set into bricks and allows the diffusion to grow from one brick to another without significant overhead.

2.2. Volume segmentation

Early volume segmentation tools used a single threshold value as seen in the interactive level-set visualization algorithm by Lefohn et al. [17]. Lundström et al. [18] introduced the α-histogram, which amplified data ranges corresponding to spatially coherent materials. They applied the method to magnetic resonance angiographies for vessel detection. Leveraging a precomputed volume tracking the feature size, Hadwiger et al. [19] presented a method for interactive exploration of volumes for feature detection and quantification. Lindholm et al. [20] estimated the material distributions in dual-energy CT data by weighing local neighborhoods of the data against approximations of material likelihood functions. They used the information to enhance transfer-function design to focus on clinically important aspects. We designed cell segmentation methods extending the Synthetic Brainbows Wan et al. [21]. Briefly, the Synthetic Brainbows technique randomly colorizes connected structures in a volume data, mimicking the results from its counterpart from bioengineering research [22]. The Synthetic Brainbows technique leveraged a cellular automaton for segmentation [23], which was implemented as computational kernels on GPUs. Specifically, the random asynchronous behavior of the GPU implementation was utilized to accelerate convergence. It was also called chaotic relaxation in computing, which theorized the necessary and sufficient conditions for an asynchronous and chaotic process to reach an expected solution [24]. For segmenting cells in a volume data, integer IDs were shuffled and uniquely assigned to every nonempty voxel. These IDs were then iteratively merged based on a series of criteria, which were designed to separate structures by stopping ID merging at structural borders. The calculation for ID merging is similar to the diffusion for brush selection. Therefore, to support large data with multiple bricks, we applied the Synthetic Brainbows to every brick and then stitched them to consolidate the result.

2.3. Biological data analysis

Tools commonly used in biomedical research for data analysis include ImageJ-Fiji [25,26], Amira [27], and Imaris [28]. With an increase of volume data size in research, the ability to quickly locate important structures and make quantitative assessment became highly desirable. Traditional tools are insufficient in terms of interactivity and large data support. There has been much work on large volume visualization and introducing the resulting interactive tools to the biomedical research community. Out-of-core ren-

dering was independently proposed by Gobbetti et al. [29] and Crassin et al. [30]. Fogal et al. [31] demonstrated that the actual amount of visualizable data in typical volume rendering is bound considerably lower than the GPU memory, and the major limitation of most rendering approaches is the inability to switch sampling rate quickly. Meyer et al. [32] used hierarchical rendering for real-color biomedical image data. Agus et al. [33] developed an interactive visualization system for medical data such as CT, MRI, and PET scans. Jeong et al. [34] visualized large-scale biomedical image stacks using display-aware processing and GPU-accelerated texture compression; large-scale neuroscience data sets were visualized and analyzed with tools called SSECRETT and NeuroTrace [35]. Fogal and Krüger [36] developed the ImageVis3D system, which used an optimized out-of-core, bricked, level-of-detail data representation to visualize large medical data. Hadwiger et al. [37] presented a visualization system that scaled to petascale volumes from high-resolution electron microscopy streams. Solteszova et al. optimized the filtering on streamed volume data to prioritize voxels with high contribution to the final visualization [38]. However, the divide-and-conquer methodology for large data visualization often poses challenges to quantitative measurements, especially when interactive data analyses are demanded. We addressed several commonly used measurements, including intensity distribution, shape, and colocalization. Our work suggested that a framework for computational modules to be easily and seamlessly assembled into a workflow was needed.

3. Methods

The description of methods generally follows a typical workflow for immunohistochemistry [39]. First, we segmented individual cells using an enhanced version of the Synthetic Brainbows framework. Then, we used brushing to select relevant cells or correct the automatically generated segmentation results. Finally, we obtained the detailed information about intensity distribution on each cell. Biologists leveraged both qualitative evaluations and quantitative measurements to identify and classify cell types. Our implementations emphasize interactivity and large data support, so that users can enjoy a fluid experience and make interactive analysis directly on visualization results without following a specific order of procedures.

3.1. Synthetic Brainbows enhanced for cell segmentation

The original Synthetic Brainbows framework included scalar-intensity and gradient-magnitude cutoff criteria for merging shuffled IDs into separate structures [21]. These criteria became insufficient for data acquired from many exploratory experiments, as the image clarity may need improvements in early-stage protocols. However, the Synthetic Brainbows framework provided plenty of freedom for the type and amount of cutoff criteria. For each iteration in the ID merging process, a voxel is checked against a series of criteria to determine if an ID merge with its neighbors occurs. We added two more criteria and provided new adjustment settings to better handle cell segmentation in a broad range of situations where multiple cells may have obscure borders.

Criterion 1: Density field. The density field is computed based on two commonly used image processing routines: adaptive histogram equalization and density-based clustering [40]. We derive three scalar fields from the original scalar intensity in two steps. **Step 1:** the original scalar field (Fig. 1a) is low-pass filtered to generate a blurred version. For rapid processing, we chose a box filter whose window size is an adjustable parameter. The resulting scalar field is defined as I_{df} (Fig. 1b). **Step 2:** we compute a local intensity distribution for each voxel of I_{df}. For the sake of simplicity, we estimate a Gaussian distribution in the intensity domain instead of a complete histogram. Hence, we compute the mean and standard deviation on I_{df}. To further accelerate the computation, we adopted an approximation approach used for the adaptive histogram equalization [41]. We divide I_{df} into regularly sized blocks and compute the mean and standard deviation only once for each block (Fig. 1c). Then, we obtain the mean and standard deviation of each voxel by bilinearly/trilinearly interpolating among the neighbor blocks. The block size for interpolation is an adjustable parameter. We define the resulting scalar fields I_{mean} and I_{std} for the mean and standard-deviation fields respectively.

Similar to the clustering methods based on density, a voxels membership in the Synthetic Brainbows ID merging process is inferred by comparing its density field value $I_{df}(\mathbf{x})$ against its local density distribution, described by $I_{mean}(\mathbf{x})$ and $I_{std}(\mathbf{x})$, where $\mathbf{x}$ is the spatial coordinates of the voxel. Specifically, a cutoff value is adjustable to stop ID propagation when the voxels density value falls outside of its local density distribution (Fig. 1d). The density-field criterion is useful for separating structures in data sets of uneven signal intensities, which can be results from insufficient penetration of fluorescent stains and occlusion of laser beams for fluorescence microscopy data.

Criterion 2: Distance field. Whereas the density-field criterion aims to segment when structural boundaries are still identifiable as connected and low-intensity signals in the density distribution, the distance-field criterion handles over-saturated signals, when structural boundaries need inference from morphological information. We compute a scalar field whose values represent the distance of a voxel to a global structural boundary. A two-step process is also employed to compute the distance field. **Step 1:** the original scalar field is low-pass filtered and binarized (Fig. 1e-f). Both the filer windows size and threshold for binarization are provided as adjustable parameters. We call the boundary of the structure in the binary image the global boundary, from which distances of inner voxels are computed. **Step 2:** in an iterative process, the voxels on each new boundary obtained from successive morphological erosion calculations are assigned with an integer of ascending order as the distance (Fig. 1g). Specifically, we start with the background as 0*s* and the global boundary 1*s*, and assign each subsequent boundary from morphological erosion with 2, 3, *etc*. The total number of iterations to generate the distance field is adjustable according to structure sizes. A standard thresholding on the distance field is sufficient to separate fused structures, whereas the threshold value is user adjustable (Fig. 1h).

We compute the morphological erosion using a 4-connected neighborhood in 2D and 6-connected neighborhood in 3D. The resulting distance field is an approximation, which computes distances based on the iteration number. Nevertheless, the distance-field criterion is sufficient for stopping ID merging and segmenting different types of cellular structures. Notably, we often combine it with other criteria in the Synthetic Brainbows framework.

Our tool provides five criteria for ID merging including three existing criteria in the Synthetic Brainbows framework: scalar intensity, gradient magnitude, density field, distance field, and size constraint. The user interface of our tool provides independent controls to enable or disable each criterion other than the scalar intensity thresholding. When only the scalar-intensity criterion is available, the most basic settings have two adjustable parameters; when all criteria are enabled, users can adjust 12 parameters according to characteristics of biological structures. Our experimental user interface started the Synthetic Brainbows calculations after a user clicked a button. So, a user first adjusted the parameters, clicked a button, and waited for the result. If further adjustments to the parameters were needed, which was often the case, the user clicked the button again and waited for the updates. This turned out to be a cumbersome process, as users often did not fully understand the meaning of each parameter and felt frustrated about

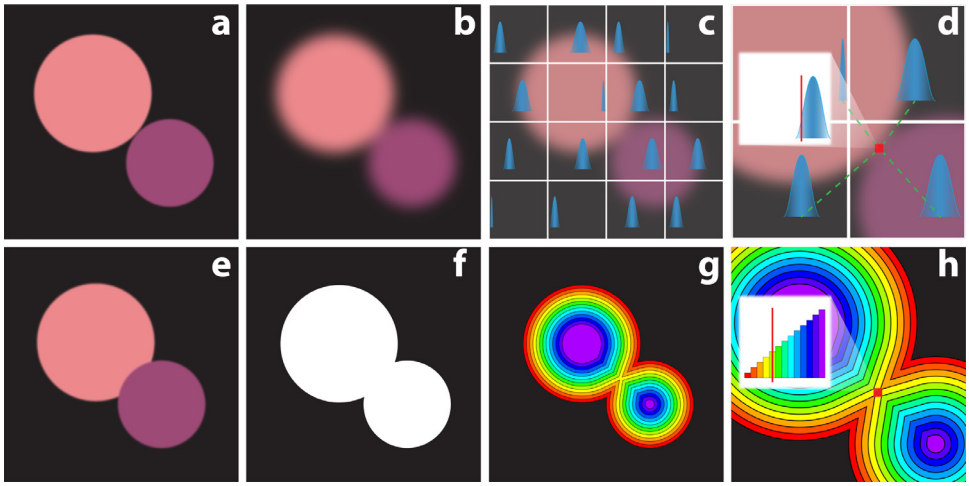

Fig. 1. New criteria in the Synthetic Brainbows framework for cell segmentation. (a) Two objects in proximity are separated using the density-field criterion. (b) Low-pass filtering result. (c) The data set is divided into blocks to obtain local density distributions. (d) A voxels density distribution is interpolated from neighbor blocks and compared against its density to stop ID merging and separate structures. (e) Two objects fusing at the border are separated using the distance-field criterion. (f) A binary image is computed by filtering and thresholding. (g) A distance field is generated on the binary image. (h) A proper thresholding on the distance field stops ID merging and separates the structures.

finding an effective configuration. Therefore, we changed the design to evaluate the Synthetic Brainbows whenever a parameter is changed via the user interface for an intuitive user experience. For example, a user drags the threshold slider of the scalar-intensity criterion and observes that the segmentation result updates in the render view. For small data sets, we achieved parameter adjusting at an interactive speed, which allowed users to learn and become fluent with the settings quickly.

We compute the Synthetic Brainbows locally to maintain interactivity for large data. Users are able to define a region of interest by brushing, which adopted the interactive segmentation method by Wan et al. [15]. The dual-stroke brush generates or updates a mask to register selected voxels as users paint on the 3D render view. The mask is a scalar volume, in which higher intensities indicate more certain selections and attenuate on structural borders for smooth selections. Each Synthetic Brainbows criterion is implemented in two GPU kernel code versions: for small volumes, the entire data set is processed at once; for large volumes, only voxels marked by nonzero mask values are processed. When the mask code is used, unprocessed voxels maintain their Synthetic Brainbows IDs to support the use case when multiple masks are sequentially applied to segment different regions. Therefore, we support an interactive segmentation workflow as follows. A user first paints directly on the render view to select structures of interest, and then adjusts the parameters of the Synthetic Brainbows while observing the changes in the resulting visualization. When corrections are needed or more structures are to be segmented, the user paints and adjusts the parameters again. The combination of the Synthetic Brainbows criteria and 3D brushing, all implemented on the GPU, provides a means to perform detailed analysis on biological structures, such as individual cells.

3.2. Volume brick support for diffusion-based brushing

The success of our interactive segmentation workflow depends on its computing speed, which is constrained by both data size and the GPU computing power. To maintain interactivity for a broad range of use cases, the segmentation and analysis in our tool employed a streamed processing approach similar to that for large volume visualization. For one, it is unrealistic to require all users to use the latest and most powerful computing hardware. For another, high-resolution acquisition techniques in fluorescence microscopy generate data that cannot be entirely processed at an interactive speed even when the most powerful consumer-level hardware is used. We adapted the streamed visualization tool developed by Wan et al. [42], which partitioned and processed a large volume data set in a stream of data bricks. Our design leverages the selection masks from brushing to constrain data analysis within the minimum number of bricks, so that the small portion of a large volume is processed at an interactive speed. This section focuses on brushing and presents methods that enable the propagation of diffusion flow to cross brick borders while maintaining the interactivity of the brushing operation itself on large data.

Using the settings of Wan et al. [42], the data partitioning grid is configured before loading a volume data set. The resulting bricks from partitioning are stored in a linear data structure whose order is updated according to the bricks priority in rendering. We added a flag for each brick to indicate the association with a selection mask from brushing. To initialize, all mask flags are set to OFF. We keep track of the content of the selection masks and update a flag to ON when an empty mask becomes selected. The paint brush tool has an inner brush stroke for seeding and an outer one for diffusion [15,16]. Therefore, we also update the mask flags within the two steps of seeding and diffusion.

Step 1: Search and flag bricks covered by seeding stroke. We start seeding and diffusion calculations only on bricks covered by the inner stroke when a user paints on the view plane (Fig. 2a). We consider that each pixel of the seeding stroke emits a ray into the volume and use ray-box intersection detection to determine brick coverage (Fig. 2b). Therefore, we flag a brick as selected when at least one ray from the seeding stroke intersects its bounding box. A complication to computing the result is that there are an indefinite number of rays emitted from an arbitrary shape from freehand painting. Our implementation leverages GPU parallelization, where each computing thread checks one pixel in screen space. When a view-plane pixel is painted by the inner stroke, the thread generates a ray based on the render view transformation and checks the ray's intersections against all bricks within the render view. We flag a brick for selection when its bounding box is intersected by at least one ray. We obtain all flagged bricks by processing all screen-space pixels in this manner. Only the flagged bricks are processed subsequently for seeding to constrain the computational load. When users keep using brief strokes for selecting cellular structures, usually a very small number of bricks are involved.

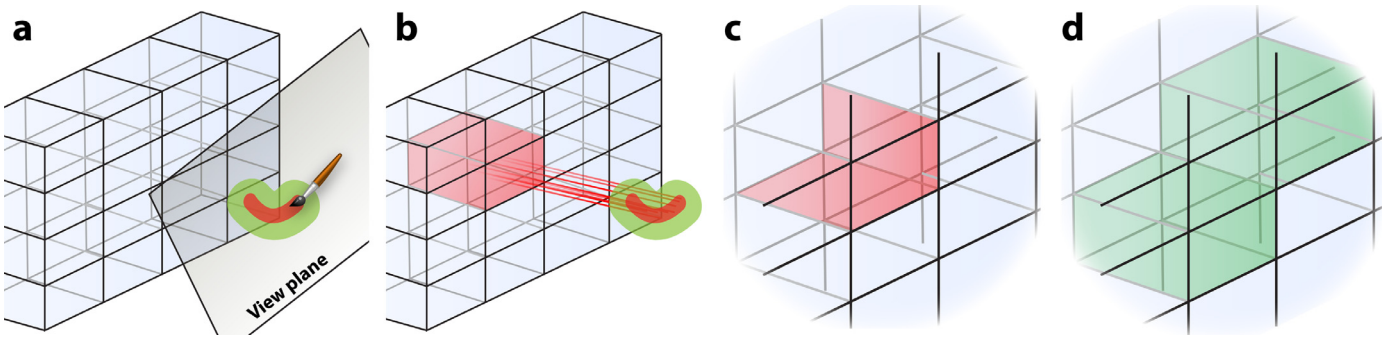

Fig. 2. We use ray-box intersection and brick face checks to include only the necessary bricks for diffusion calculations. (a) User paints on the view plane to select 3D structures from a volume data set, which is partitioned into bricks for streamed processing. The selection brush paints two strokes simultaneously. The inner stroke (red) is for seeding and the outer (green) for diffusion. (b) The screen pixels of the inner stroke emit rays to detect the bricks for seeding. The red brick is flagged for seeding because it is the only one intersected by the rays. (c) In an iteration of the diffusion calculation, we check the faces (red) of the flagged brick for diffusion flow crossing. We copy the brick face voxels to its neighbor when the diffusion flow reaches the faces. (d) We flag the neighbor bricks (green) for diffusion calculation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Step 2: Flag neighbor bricks during diffusion flow propagation. Seed voxels selected by the inner brush stroke propagate using the morphological diffusion [15], which is constrained by the larger outer stroke. Because we only compute diffusion on flagged bricks, new bricks reached by diffusion flow during propagation need to be flagged. Here, we adopted a different approach than checking ray-box intersections in Step 1. In an iterative process of the morphological diffusion, we check the faces of each flagged brick for each iteration. We define the *face* of a brick as the voxels in the outermost slice of thickness 1. Two neighbor bricks share one face: the original volume data are duplicated at brick faces, which allow certain calculations such as the gradient magnitude to be seamless over brick borders. Similarly, selection masks are duplicated at brick faces to enable diffusion flow crossing. When a brick face in the mask volume contains at least one voxel marked as being selected, all voxels of the face are copied to its neighbor brick, which is then flagged for selection (Fig. 2c-d). Checking and copying the face voxels are parallelized on GPU as separate image-processing kernels. Compared to flagging all bricks covered by the outer brush stroke at once using the method in Step 1, checking brick faces saves the computing time for diffusion calculations when they are necessary. Furthermore, knowing exactly when to copy face voxels across bricks makes efficient data exchanges. The actual number of iterations to complete the diffusion calculation is user adjustable and usually requires a value around 20. Considering that the chance for a diffusion flow to reach and cross a brick border in an iteration is low, we further reduced the computing time by checking fewer brick faces. We only check 2 of 6 brick faces for each iteration and rotate the checks in every 3 iterations. For example, we name the brick faces according to the coordinate axis a face is perpendicular to, i.e., +X, -X, +Y, -Y, +Z, and -Z. The brick face checks over multiple iterations are: Iteration 1: +X, -X, Iteration 2: +Y, -Y, Iteration 3: +Z, -Z, and repeat. If we happen to miss a face check using this order, the delay of its copy is 2 iterations at most.

For selecting relatively small structures such as cells from a large data set, our implementation of the brush tool for streamed processing is able to maintain interactivity, because the number of data bricks that need expensive calculations is constrained. Users need to configure the brick size according to their system's computing power, which may need some experiments and getting used to. The results from brushing include a mask volume marking selected voxels, which shares the same partitioning grid with the original data, and a series of flags to determine if each brick is selected. When we combine these results with the enhanced Synthetic Brainbows in Section 3.1, we complete the interactive brushing-segmentation workflow for large volumes by constraining the workload to only the flagged bricks and masked voxels. Therefore, our design achieved the goal of interactive segmentation for large data based on user guidance. The user interactions are also intuitive: a user first uses the paint brush tool to select a portion from a large volume, and then adjusts the sliders for the Synthetic Brainbows parameters to visually and interactively confirm the segmentation results.

3.3. Brick stitching

The segmentation results from Section 3.2 are disjointed when multiple data bricks are present. For both visualization and quantitative analysis, the Synthetic Brainbows results need stitching. Firstly, users may want to apply the parameters obtained from partially selected regions to an entire data set. We provide a practical and straightforward solution that removes partitioning from original data and automatically applies existing parameter settings for segmentation. The same method is also applicable as a batch process for time-dependent (4D) data or data from multiple experiments. However, this method is time consuming for large data and difficult to use for handling variations within or across data sets because, secondly, users may just need rapid analysis results on certain structures under a focused study. Therefore, methods to stitch multiple bricks at an interactive speed are desired.

In the Synthetic Brainbows results, we use integer IDs to distinguish structures within a single data brick. The algorithm that shuffles and then merges these integers indicates that a segmented structure can be assigned with any integer. When multiple bricks are present, the Synthetic Brainbows are computed independently. To uniquely identify a structure in segmentation results across multiple bricks, we use the Synthetic Brainbows ID and a brick ID jointly. We name this ID combination the extended ID, or EID, which is available for either flagged bricks from brushing or all bricks when an entire data set is processed. To stitch bricks, we examine every brick neighbor pair and check the shared brick face to stitch EIDs of contacting structures. A hurdle to a straightforward implementation of EID merging is noise because they are meaningless structures potentially consuming a significant amount of memory. The Synthetic Brainbows method initializes an ID volume with shuffled integers that uniquely identify individual voxels. Not all IDs are merged in the process: low-intensity background is filled with disjointed IDs; IDs of noise signals merge into minuscule groups of very low voxel counts. The uncertainty about the number of EIDs to stitch makes it difficult to allocate memory on the GPU. We leveraged the size-constraint criterion of the original

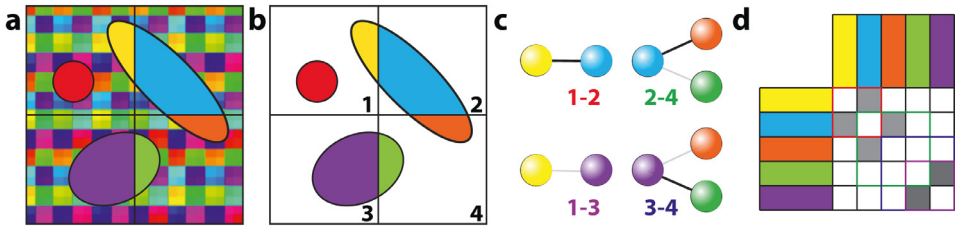

Fig. 3. The Synthetic Brainbows are computed independently in data bricks and then stitched. (a) Resulting IDs after Synthetic Brainbows. Each structure in a brick is labeled by an integer ID, which is visualized as a distinct color. (b) We clean the result according to structure size, removing small structures and disjointed IDs. (c) We compute the connectivity between structures for each brick neighbor pair. A bipartite graph is generated for each pair, where a solid line denotes connection at a brick face. (d) Each graph is stored as an adjacency matrix in the GPU memory and assembled into an EID graph in the CPU memory. The scalar values of the graph nodes represent contacting sizes, which are used to prune the EID graph.

Synthetic Brainbows and stitched EIDs from multiple bricks using 3 steps: size-based data cleaning, EID graph building, and graph pruning.

Step 1: Size-based data cleaning. We separate meaningful structures from background and noise based on their voxel counts (Fig. 3a-b). Retrieving the voxel count for each unique ID was a feature in the original Synthetic Brainbows and used as the size-constraint criterion to control ID merging [21]. Briefly, a temporary buffer of the same size as a brick is used to count IDs. Although an ID may migrate anywhere in a brick during merging, its origin in terms of the XYZ coordinates is uniquely determined by the inverse of the ID shuffling function. The origin in the count buffer is then used to register the count of this ID as we peruse the ID volume. Essentially using the shuffling function as an index table, different IDs are thus counted in parallel. IDs of meaningful structures are obtained by thresholding the count buffer, whereas the threshold is an externally controlled parameter. This implementation avoided the difficulty of handling dynamically allocated memory on the GPU because of the unknown count of meaningful structures beforehand. We only need to count the values of the thresholded count buffer to obtain the number of meaningful structures. Subsequent memory allocations on the GPU becomes straightforward.

Step 2: EID graph building. A bipartite graph is used to describe the relationships of contacting EIDs at a brick face (Fig. 3c). Again, to avoid dynamic memory allocation, an adjacency matrix is used to store the graph. The value of an element in the matrix is the count of contacting voxels between two EIDs, which are represented as the row and column indices. We only need a lookup table for the conversion between EIDs and row/column indices to populate the matrix. Similar to the face checks for brushing, our GPU code simultaneously reads two slices of IDs from two neighbor bricks colocated at the brick face. When two colocated IDs are nonzero and exist as row/column indices of the adjacency matrix (checked through the lookup table), we increase the value of the corresponding matrix element by 1.

Step 3: Graph pruning. The resulting adjacency matrices contain EID contacting information at brick faces. The value of a matrix element describes the degree of contacting between two structures. A complete EID graph includes contacting information for all processed bricks. Considering that one structure may span more than two bricks, the final EID stitching needs to find all connected components on the graph. We read back the adjacency matrices from the GPU memory to the CPU memory and assemble them into a complete EID graph. The size of an adjacency matrix for contacting structures on a brick border depends on both structure size and brick size. For an analysis of cells, the number of border-spanning structures in a matrix is often small, up to $10-20$, when a typical brick size setting around 256 is used for a cell diameter of several tens of voxels. The assembled EID graph is sparse and can be considerably larger, containing several thousand cells. Therefore, we leveraged the Boost Graph Library and used the linked lists to store and process the assembled graph [43]. A nonzero element on the EID graph indicates contacts between two EIDs (Fig. 3d). The EID graph is pruned according to contacting size, because we do not want to stitch EIDs that are slightly connected.

To summarize, applying an interactive brushing-segmentation workflow on a multibrick scalar volume, we obtained derived data as follows: a multibrick mask volume to identify brush-selected structures, a multibrick ID volume to segment distinct structures, a series of flags to identify processed bricks, and an EID graph to stitch structures among bricks.

3.4. Data analysis in streamed processing

Leveraging the data derived from the interactive brushing-segmentation workflow, we provide a series of quantitative analyses at an interactive speed. Users can choose to perform analysis on an entire volume, partial volume marked by a selection mask, or grouped EIDs from segmentation. We describe these analysis functions with regard to segmented structures because that is the most complex case to implement. We have obtained all needed information to constrain the calculations to segmented structures within a multibrick volume so that quantitative analysis becomes interactive. The supported analyses are as follows.

Size. ID counting in a brick, which was discussed in Step 1 of Section 3.3, is the basis for most analyses that compute results by accumulating values from voxels. Leveraging the EID graph, we obtain the voxel count of a segmented structure by summing up the ID counts in multiple bricks. Once the voxel count of a structure is obtained, its physical size is computed as the product of the count and the physical size of a voxel, which is either provided by the metadata of a fluorescence microscopy scan or user defined. A pitfall in reading the size information directly from voxel count is that a structure's size value changes significantly when a large number of low-intensity voxels can be included or excluded during brushing or segmentation without much user awareness. Many factors influence the membership of a low intensity voxel in an interactive analysis workflow, including brushing operations, segmentation settings, and visualization adjustments. We provide the intensity-weighted size, which accumulates the normalized values (intensity values mapped to $[0,1]$) instead of ID counts, to reduce the influence of low intensity voxels and enable meaningful comparisons. Its computation is similar to voxel counting, but an additional buffer for accumulating intensity values is needed. These accumulation calculations (IDs and intensity values, as well as subsequently discussed analyses) can be performed simultaneously in a single pass.

Center location. We compute the center coordinates of a structure by accumulating coordinates of voxels and dividing it by voxel

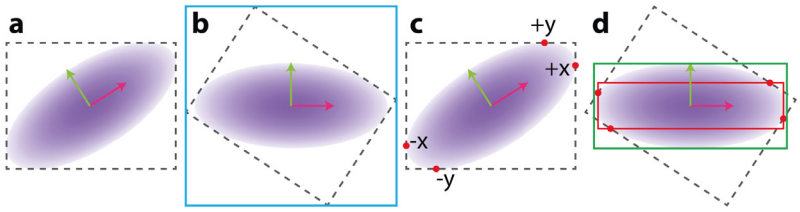

Fig. 4. Bounding box is insufficient to compute the extent of a rotated structure. (a) We compute the principal components (red and green vectors) and bounding box (dotted lines) of a Gaussian distribution. (b) The distribution is rotated to align the principal components with the spatial coordinates. The new bounding box (blue) is oversize. (c) We use four bounding points (red) contacting the original bounding box to describe its extent. (d) The new extent (red) from the bounding points is a better approximation to the true bounding box (green) after rotation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

count. The influence of low intensity voxels to the computing of centers is less obvious, especially for cellular structures, because the spatial distribution of voxel intensity in a fluorescence microscopy scan can be considered symmetric in general, as high intensity structures attenuate into background at similar rates in all directions. However, if intensity-weighted center coordinates are required, the accumulation can be computed similarly as the intensity-weighted size.

Intensity distribution. We compute four entries to describe the distribution of scalar intensities in a segmented structure: mean, minimum, maximum, and standard deviation. The mean intensity is computed from the accumulated intensity ($I(x)$) and voxel count (N): $\bar{I} = \sum I(x)/N$. The minimum and maximum values are computed similar to the accumulated intensity, but additional buffers are needed. We derived an equation for the standard deviation for streamed processing because the textbook formula needs the mean within a summation, which we do not know beforehand. To compute the standard deviation in one pass, we separate the mean from $I(x)$:

$$I_{std} = \sqrt{\frac{1}{N-1}\sum_{x=1}^{N}(I(x)-\bar{I})^2} \tag{1}$$

$$= \sqrt{\frac{1}{N-1}}\sqrt{\sum_{x=1}^{N} I(x)^2 + N\bar{I}^2 - 2\bar{I}\sum_{x=1}^{N} I(x)} \tag{2}$$

Consequently, we compute $\sum I(x)$ and $\sum I(x)^2$ independently and plug in $\bar{I}$ and N later when they become known. The computing of $\sum I(x)^2$ is the same as $\sum I(x)$, which needs a separate buffer.

Principal components. We included the principal component analysis (PCA) for the 3D orientation of a segmented structure [44]. The covariance matrix of the voxel coordinates is computed in a single pass using the same method as the standard deviation. Supposing the coordinates of a voxel is (x, y, z), we accumulate these values independently: $\sum x^2$, $\sum y^2$, $\sum z^2$, $\sum xy$, $\sum xz$, and $\sum yz$. The covariance matrix is then computed after the mean coordinates become known. The eigen decomposition of the covariance matrix gives us three vectors indicating the spatial orientation of the structure. To compute the extent of the structure in the new orientation, we leveraged the idea of a bounding box to avoid parsing through all voxels again. Computing the extent directly from a rotated bounding box is obviously inaccurate (Fig. 4a-b). Instead of the corners of a bounding box, we want to use points from the structure and sample as many as possible to approximate its true shape. A good approximation was achieved using 6 points contacting the original bounding box (+x, −x, +y, −y, +z, −z, Fig. 4c). These "bounding points" are rotated to compute a new bounding box, which is closer to the true bounding box (Fig. 4d). We consider the result a good trade-off for its negligible overhead and accuracy on the major axis.

Colocalization. The colocalization analysis is a quantitative measure of the spatial relationships of intensity distributions among multiple channels. Specifically, we count the number of voxels from two channels that coincide and are both above a threshold value. Its implementation on the GPU is similar to the voxel count in a single channel, except that two volumes are sampled simultaneously. In light of that the intensity-weighted size often provides a robust measurement when brushing is employed to isolate structures (refer to size analysis), we provide three methods for colocalization analysis. 1) *Threshold-Coincidence* computes the overlapping regions by first thresholding the channels, and then counting the overlap as the voxels in both channels that have intensity values above a threshold. 2) *Min Value* accumulates the minimum intensity (normalized to $[0, 1]$) from two channels at each voxel. 3) *Product* accumulates the product of intensity values (normalized to $[0, 1]$) from two channels at each voxel. For more than two channels of volume data, we evaluate the colocalization in pairs and compose the pairwise results in an adjacent matrix. The presentation of the resulting matrix can be further customized for easy comparison. 1) *Ratio* displays each element of the matrix as a percentage value for the ratio between matrix element (r, c) and (r, r), where r and c are indices of rows and columns. 2) *Intensity-Weighted* displays the intensity-weighted voxel counts. Therefore, the Manders' coefficients [4] are computed by enabling *Threshold-Coincidence*, *Ratio*, and *Intensity-Weighted*. 3) *Physical Size* displays the physical size using the voxel size information from metadata. 4) *Color Map* displays the matrix cells in colors, which are configured in volume channels' visualization settings.

3.5. A framework for rapid analysis

A complete workflow for analyzing fluorescence microscopy data is assembled by concatenating the functions that we have presented (Fig. 5a). The fluidity of operations to end users depends on the combined processing time when all modules are executed: users paint to select structures from the render view, adjust parameters afterward, and expect to see the analysis results immediately on the selection. For standalone modules, we already presented computational details that enabled GPU code running at an interactive speed. We further developed a programming framework to manage multiple GPU programs and memory objects, providing an easy-to-use interface by grouping API calls and an efficient workflow by reducing memory exchanges in back-to-back executions. Our framework also provides integrated support for the data bricks in streamed processing, i.e., the execution of GPU code automatically loops through bricks flagged for processing.

We used OpenGL and OpenCL to develop the interactive data analysis tool, leveraging OpenGL-OpenCL interoperations for resource sharing on GPU. Data analysis methods presented in preceding sections were implemented as OpenCL kernel programs. We designed a KernelProgram class to manage the code for standalone analysis routines, which included features as follows (Fig. 5b). 1)

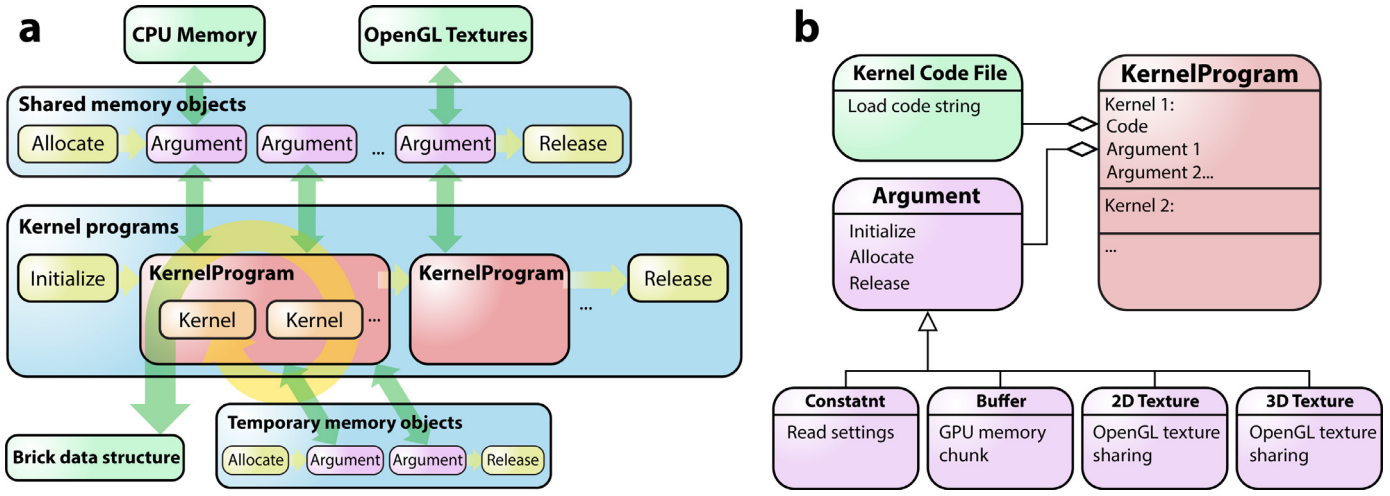

Fig. 5. We designed a framework to organize kernel programs and GPU memory objects for rapid analysis workflows. (a) An illustration of a typical workflow, where multiple kernel programs are used. We use the Argument class to manage memory objects and data exchange. The loops of kernel programs are controlled by the brick data structure. (b) An illustration of the relationship between the KernelProgram and Argument classes.

The KernelProgram integrates the code for system initialization. 2) A program can manage multiple kernels, which are grouped together because an analysis function usually needs multiple passes of processing or different settings for an analysis are implemented in multiple kernels. The actually executed kernels as well as their execution order are controlled via settings externally. 3) The KernelProgram is executed in a loop for either an iterative process or multiple data bricks (Fig. 5a). The execution loop is driven by the data structures for bricks and their selection flags. Non-selected or empty bricks are skipped automatically in the loop. 4) Memory management and argument assignment for the KernelProgram are grouped under a separate class, called Argument. Each kernel argument of the OpenCL code maps to an Argument object, which in turn maps to local memory objects, CPU memory, or shared OpenGL resources. Sharing memory objects among kernel programs on GPU is important to reduce data exchange and executing time. We further detail the design of the Argument class as follows.

We designed the Argument class to manage OpenCL memory objects and map them to kernel arguments (Fig. 5b). We aimed at an efficient memory management for GPU computing to minimize the number of operations for allocation and duplication, including copying data from CPU memory to GPU memory, reading data back from GPU memory to CPU memory, and allocating duplicated data on the GPU. We designed the Argument class to automate the memory management scenarios as follows. 1) **Constant.** Constants are parameter settings passed from CPU memory. 2) **Buffer.** We use buffers for temporary storage or results to be read back to CPU memory. A buffers contents are initialized using a kernel or copied from CPU memory. More importantly, a buffer can be shared by multiple kernels to avoid data exchange with the CPU. The initialization and release of an Argument object are controlled independently of that of a kernel, which allows buffers to pass among kernels. However, we also organize kernels sharing temporary buffers under one KernelProgram, so that the temporary buffers can be properly released after all data bricks are processed amid a workflow (Fig. 5a). 3) **2D texture.** The Argument class also manages resource sharing with OpenGL, including reading 2D textures generated in visualization. For example, the paint brush creates a 2D texture object for the render view framebuffer. The computing of ray-box intersections reads this 2D texture as an argument. 4) **3D texture.** Both original and derived volume data are stored as 3D texture objects, including data bricks, selection masks, and ID volumes.

Table 1
Summary of performance benchmarks.

Time (Sec.) / Brick size*	Full**	2048	1024	512	256	128
Paint selection	1.1541	0.8537	0.4337	0.3400	0.3196	0.4214
Segmentation	N/A***	11.3171	3.1100	0.8188	0.2654	0.0488
Colocalization	2.8404	1.9349	0.3332	0.0791	0.0259	0.0043

*The maximum voxel count in any spatial dimension of a brick. **Undivided volume as a single brick ($3179 \times 2047 \times 181$ voxels). ***Program crash.

4. Results

The methods for interactive analysis of cellular structures in large volume data were implemented as modules in the FluoRender system, which is an open-source project at https://github.com/SCIInstitute/fluorender. The interactivity for analyzing large volume data was validated by a series of speed benchmarks detailed in the supplementary materials. Table 1 summarizes the performance results. Our implementation allowed the computing time to track the data size linearly for commonly used brick sizes. Therefore, interactive data analysis was achieved on computer systems with varying performance levels when large data were partitioned and only the subset of interest was processed at a time.

We applied the tool for the identification and classification of progenitor cells in the mouse inner ear. Most of the methods presented here were the result from the seminal work that needed intuitive and rapid data analysis for 3D fluorescence microscopy. For details about the biological background, experiment protocols, and scientific significance, please refer to the publications by Holman et al. [39,45]. Briefly, the biologists investigated transgenic cells in the vestibular and auditory systems. A transgenic mouse specifically enabled the observation of progenitor-like cells organizing into rosettes in the *eminentia cruciatum* (*EC*) of vertical semicircular canals. The identification of previously uncharacterized cells shed light on their roles for the development and repairing of inner-ear mechanosensory structures. Our tool was used to examine a large amount of confocal microscopy data consisting of various ages and cell stages in whole mount fixed tissues. To locate and portray individual progenitor-like cells, several hundred cells in a single tissue sample were segmented and classified according to morphology. Common in exploratory studies where vaguely characterized cells are being investigated, an investigators analysis based on visual examination and assisted by immediate quan-

titative evaluations played a crucial role in determining cell types. The computational modules presented here were assembled into a workflow for researchers to make quick classification and reliable comparison.

In the study, mice of either sex at early postnatal ages were used. Images of epithelia from anterior crista were acquired using confocal microscopy. Our tool visualized multiple channels of volume data from confocal microscopy using interactive volume rendering. Before any quantitative analysis, we enhanced the visualization for each channel using rendering properties including saturation, luminance, Gamma, and threshold [46,47]. A clarified visualization of the cells allowed researchers to focus on the most important part of a data set. More details about the intensity, size, and morphology were first obtained on select cells using brushing. We enabled the paint brush tool and moved the mouse cursor into the 3D render view. The stamp of the paint brush was indicated by two circles, which were for seeding and diffusion, respectively. We clicked and dragged the mouse within the 3D render view and made sure the brush stroke covered a cell whose information we wanted to retrieve. We released the mouse button to finish the brush stroke as well as the selection in 3D. A satisfactory selection on the cell could often be obtained because we considered factors including intensity, gradient, and occlusion when the diffusion was evaluated within the projected region of the brush stroke. Additionally, we were able to modify the selection interactively by adjusting threshold settings and brushing from a rotated viewpoint to append or erase selections. Quantitative analysis results were immediately obtained for the selection.

When a group of cells were selected, we used the Synthetic Brainbows segmentation to identify and separate them. From the user interface, we set the segmentation to be applied only to the brush selected regions. The segmentation results were visualized in 3D as individual cells were assigned with distinct colors. When a confocal scan was noisy or the selected region contained cells obstructed by background noises, we adjusted the parameters for the Synthetic Brainbows and observed the changes interactively. We always started with the threshold parameter and made fine adjustment using the density-field and distance-field settings. Sometimes a size-constraint was applied to stop ID merging between different cells. If a satisfactory segmentation of the selected cells could not be quickly obtained by parameter adjustments, we made modifications to the selection mask by brushing. Specifically, an eraser brush of hairline size was used to separate two fused cells by erasing their borders in the selection mask. We also used the eraser brush to remove problematic cells entirely, so that the remainder could be easily segmented. We adjusted the Synthetic Brainbows parameters again to fine-tune the segmentation results on the updated selection. During this interactive process, we paid attention to the settings that were effective so that they became first choices for data analyzed later. Occasionally it was difficult to obtain settings that were universally effective to correctly segment every cell. This was where an interactive tool became instrumental for users to obtain desired results eventually. The strategies used include: 1) we visually divided one data set into several regions, used the brush tool to select each region, and then adjusted segmentation settings for each region; 2) we applied universal settings to an entire data set, making sure that most cells were correctly segmented, and then made fine adjustments to individual poorly segmented cells.

Accurate cell information was important to the study because we filtered and classified cells according to sizes. We also compared cell sizes across a series of scans at various development stages. Therefore, the interactive analysis workflow in our tool became indispensable for not only separating cells correctly, but also obtaining correct cell sizes under user guidance. After carefully examining the cell segmentation in 3D and referencing the quantitative measurements on a list, we confirmed the results and proceeded to cell classification. An initial sorting of the cells was obtained only by size. However, we obtained cell types by meticulously evaluating every cell and considered factors including morphology, fluorescence intensity, and location. To obtain correct spatial locations within the organ (crista), we kept the users oriented by rotating the scans from various development stages to a common coordinate frame. We brush selected the characteristic features, such as the apical protrusion in *EC*, and then applied PCA to obtain the organ orientation as the common coordinate frame. To make sure accurate cell classifications, each data set was cross-examined by at least two users. Fig. 6 is the result from four scans at different development ages (Fig. 6 P1-533, i.e., 1 day to 533 days after birth). We classified the cells into four categories: clino2 cells, hair cells, supporting cells, and unidentified types. Since the study focused on the unique characteristics of the clino2 cells, we also analyzed their growth over time. Detailed in Holman et al. [39], the discovery of the clinocytes and clino2 cells led to a hypothetical model that outlined the lineage and potential regenerative capacity of these cells in the mammalian vestibular neuroepithelium.

We performed precise colocalization analysis on individual cells based on results from brushing and segmentation. We copied the selection mask painted on one channel (Fig. 7b immunolabeling ATOH1) to another (Fig. 7a immunolabeling SOX2). Notice that a cell in the SOX2 channel has lower signal intensity at its nucleus. But the cell was stained in entirety in the ATOH1 channel. We were able to quantify this qualitative observation by separating a cell into nucleus and cytoplasm (Fig. 7c&d) and obtaining the colocalization measurements separately. Fig. 7g shows a series of colocalization coefficients between the SOX2 and ATOH1 channels. Without a selection mask, the result (All channel in Fig. 7g) was different from those when a single cell was selected. Obviously, the staining of the nucleus only in the ATOH1 channel yielded the ratio difference between the SOX2 and ATOH1 channels. When the nucleus was either removed or isolated, the two channels became highly coincident in the cytoplasm (Cytoplasm in Fig. 7g) and ATOH1-dominant in the nucleus (Nuc. Only in Fig. 7g). Analysis at the cellular level confirmed that the coefficients from using independent cell masks (Ind. masks in Fig. 7g) were mainly influenced by, and indicate, the uneven intensity distribution at the nucleus between the two channels. Fig. 7h shows the colocalization coefficients for 24 cells, which exhibit varying levels of coincidence between channels. Such quantitative information provided us knowledge on the different characteristics of nuclear staining for SOX2 and ATOH1 at the specific age, which was not immediately obvious from just the visualizations in Fig. 7e-f.

5. Discussion

We combined a series of techniques, from volume visualization to data analysis, and made a tool for rapid measurements on fluorescence microscopy data. Many design choices were made with an ultimate goal in mind: maintaining interactivity for large data on consumer-level computers. Unlike interactive volume visualization for large data, there are fewer leverages for analyzing large volume data. Commonly found in volume visualization tools, multiple resolution levels are generated to accelerate the search and sampling of voxels in large data. A version of the data with reduced resolution accommodates interactive viewing at low magnifications because a computer display has a finite resolution, and a human user cannot comprehend very much visual information simultaneously. There were problems applying the same approach to data analysis because the results obtained on a down-sampled version of the original data often did not satisfy the requirement for accuracy. While there are methods that leverage precomputed information about intensity distribution [48,49], they limit the types of analy-

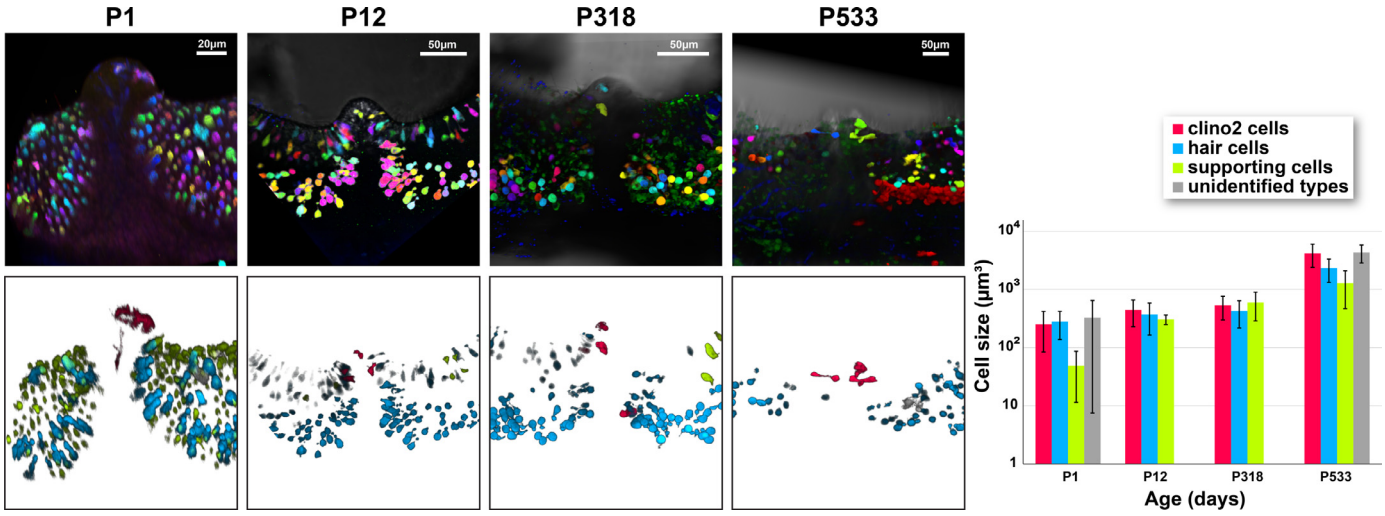

Fig. 6. Four vertical semicircular canals from mice at different development stages (P1-533) were imaged and analyzed. We segmented cells for each scan. Each cell was examined and classified into one of four categories. The top row shows the cells colored by the Synthetic Brainbows; the bottom row shows cell categories by colors.

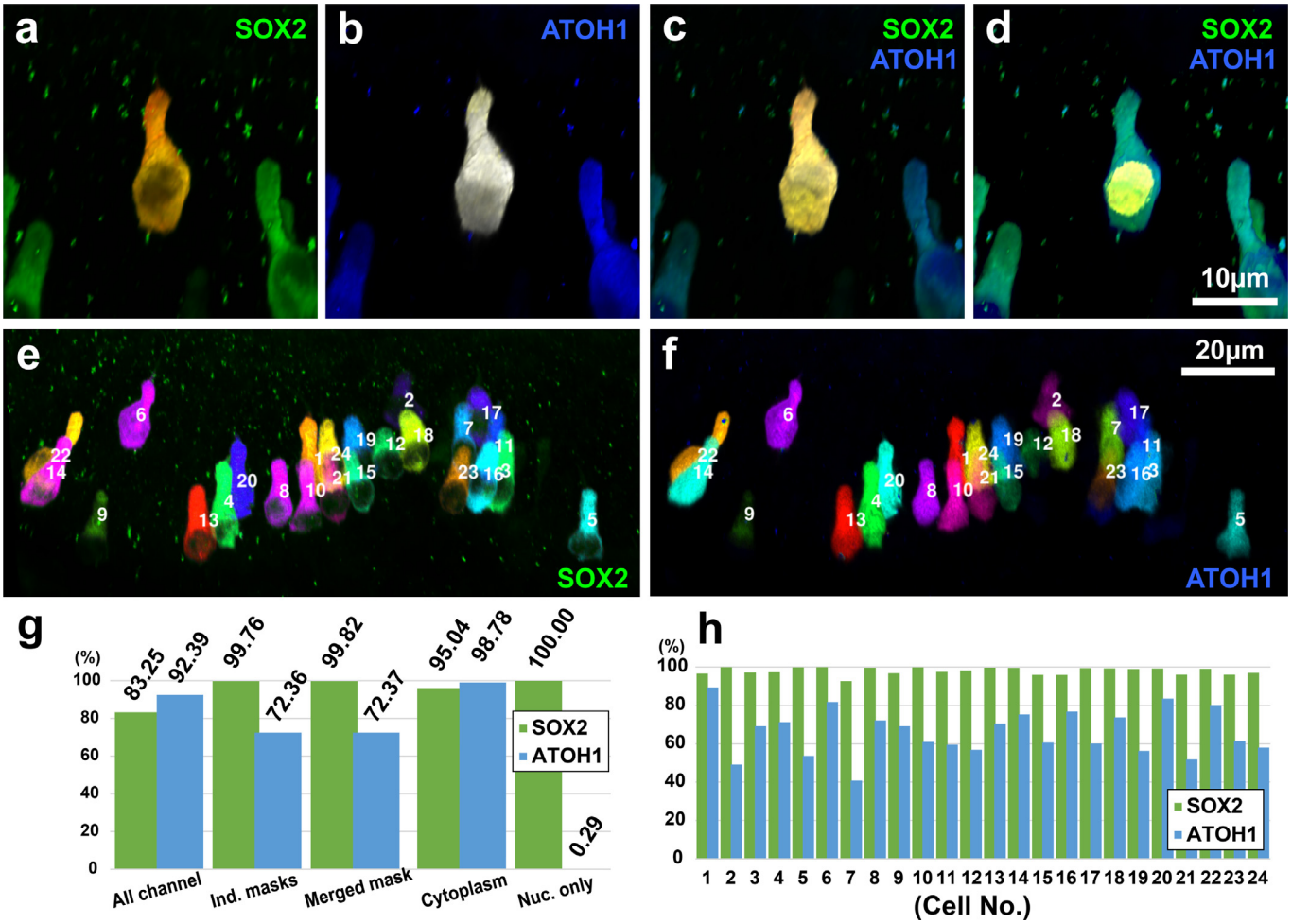

Fig. 7. A rapid cell analysis workflow allowed focused and precise colocalization analysis between two channels. (a) The highlighted selection mask did not include the nucleus in a channel immunolabeling SOX2. (b) The cell, including the nucleus, was highlighted using the selection mask in the ATOH1 channel. (c) Colocalization analysis was performed based on the selection mask excluding the nucleus. (d) Colocalization analysis was performed based on the selection mask including only the nucleus, which was obtained by subtracting one mask from another. (e) The Synthetic Brainows result. Each cell in the SOX2 channel was assigned an ID and unique color. (f) The Synthetic Brainbows result for the ATOH1 channel. (g) The quantitative colocalization analysis results of a single cell in a–d. Ratios were calculated as the Manders coefficients (the Threshold-Coincidence method with Intensity-Weighted option). All channel means that the analysis was performed on two complete channels without selection mask; Ind. masks means that selection masks were generated independently for each channel according to their structures; Merged mask means that the masks from both channels were merged and applied to both; Cytoplasm means that a selection mask excluding the nucleus was used; Nuc. Only means that a selection mask defining only the nucleus was used. (h) A detailed colocalization analysis was performed on each cell. A total of 24 cells were analyzed.

sis that can be subsequently performed. Our use case scenario demanded a tool that allowed users to visualize and analyze volume data as soon as they were acquired from a 3D fluorescence microscope for high-throughput applications. Users often needed a series of quantitative measurements to analyze data initially as they are being acquired. As a result, we decided to leverage user interactions to constrain data size to be processed in our tool. Specifically, we used brushing to isolate structures and constrain analysis to regions where users considered important. The resulting tool provided intuitive operations because its workflow resembled how we usually approach a new piece of material in real life: we browse for an overview, locate relevant parts, make annotations, take notes, and draw sketches. From a users perspective, obtaining the quantitative information on a structure can be as simple as brushing with a single stroke. Multiple computational steps were assembled behind the scene to achieve the simplicity, especially when multiple bricks were present to process large data. In practice, the interactivity can be influenced by many factors: hardware capabilities, data size, settings for brick size, and user experience. We performed several speed benchmarks to determine a proper brick size choice to maintain interactivity on a specific hardware platform. The benchmark results are in the supplementary materials. A complimentary video demonstrates significant speedup on a common consumer-level computer. Furthermore, we reported that our tool had contributed to real biomedical research and become instrumental to the identification of new cell types.

Although we placed emphasis on the efficiency of our method running on the GPU, we did not compromise the quality in our design. For example, much computing time could have been saved if the diffusion-based brush selection were replaced with simple thresholding. The morphological diffusion automatically detects structural boundaries in 3D and improves usability. So, we focused on methods that correctly flagged data bricks by detecting border crossing of the diffusion flow. Additionally, our design of the diffusion brush can be extended for unbounded growth. For example, to select a branching structure in 3D, we can set seed points at the root and then allow the seeds to diffuse freely. New bricks are flagged and added for the computations when the diffusion flow reaches brick borders. Thus, we plan to extend this work for interactive segmentation of neural structures including single neurons, neuroblasts, and nerve bundles. We will also obtain the topological information of these structures when we connect the step-by-step growth of the diffusion flow.

The Synthetic Brainbows technique provided an extensible framework that accommodated multiple criteria to stop ID merging. We added more criteria to the original framework to broaden the application for various cellular structures. Although interactive adjustment of the segmentation parameters simplified user operations and allowed the visualization updates to be quickly learned by users, further improvements are viable using certain machine-learning techniques to automate parameter adjustments. Segmentation results generated by brushing and interactive parameter adjustments will provide training examples for a machine-learning algorithm. Furthermore, we can record all the user interactions for fixing segmentation results and feed them back to the machine-learning algorithm for an online training process [50]. The trained objective function will be applied to similar data to improve efficiency.

When we analyzed multichannel data sets using the colocalization functions, the resulting ratios varied with the choice of computing method. Therefore, we chose a method depending on the characteristics of a data set. The Threshold-Coincidence method was used most often. However, setting a threshold value for each channel or structure was laborious. The brushing tool addressed this issue because thresholding was built into the brush selection calculations. The Min-Value method avoided threshold settings altogether but worked best for two channels having comparable intensity levels. Otherwise, one channel with a significantly lower intensity level predominated the colocalization. The Product method generated a voxels colocalization value between two channels on the adjacency matrix ($rr < cr < cc$, if $r < c$ and $r, c \in [0, 1]$). Therefore, the colocalization ratio sometimes exceeded 100% on the channel with a significantly lower intensity level. We leveraged this phenomenon to determine the relative intensity levels of two channels, whereas all three methods should give consistent results for channels with comparable fluorescence intensity levels.

6. Conclusion

This work presented a tool for interactive analysis of 3D fluorescence microscopy data, emphasizing the analysis of cellular structures. We combined a series of techniques and designed GPU-friendly methods for large volume data, which were partitioned into regularly sized bricks for streamed processing. The methods were assembled into an interactive workflow to obtain quantitative information of accurately segmented cells in biomedical research. We achieved our design goals, providing intuitive user interactions including brushing and slider adjustments. Quantitative results as well as linked 3D visualizations were updated according to user guidance at an interactive speed. We successfully applied the tool to whole-mount tissues as demonstrated for mouse vestibular organs. Quantitative and precise analysis of every cell in the resulting volume data would not have been practical without an easy-to-use tool, which received contributions from experts of diverse domains. The collaborative work between biologists and computer scientists was reported in a sister Elsevier journal. The future plan for our tool includes extensions to further improve usability and broaden data support. Specifically, we will combine machine-learning algorithms with user interactions to simplify parameter adjustments for various data types. We will include more quantitative measurements based on user feedback. And we will add support for various microscopy formats including large data from mosaicking and super-resolution microscopes.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Yong Wan: Conceptualization, Methodology, Software, Data curation, Writing - original draft. **Holly A. Holman:** Conceptualization, Validation, Formal analysis, Resources, Data curation, Writing - review & editing, Visualization. **Charles Hansen:** Conceptualization, Methodology, Writing - review & editing, Supervision, Funding acquisition.

Acknowledgments

Many people have contributed to the development of FluoRender through discussions or collaborations. We would especially like to thank FluoRender contributors and users that are instrumental to this work, including H. Otsuna, G. Kardon, K. Ito, B. Bagley, G. Stanfield, A. Wong, R. Goldstein, and K. Asahina. We also wish C.-B. Chien could have seen the latest development of FluoRender. He has left us too early in 2011. FluoRender was supported by NIH grants: P41 GM103545 and R01 EB023947. The study of mouse vestibular epithelia was supported, in part, by a grant from the NIDCD: R01DC006685.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.cag.2021.05.006.

References

[1] Davidovits P, Egger MD. Scanning laser microscope. Nature 1969;223:831.
[2] Denk W, Strickler JH, Webb WW. Two-photo laser scanning fluorescence microscopy. Science 1990;248:73–6.
[3] Manders EM, Stap J, Brakenhoff GJ, van Driel R, Aten JA. Dynamics of three--dimensional replication patterns during the s-phase, analysed by double labelling of DNA and confocal microscopy. J Cell Sci 1992;103:857–62.
[4] Manders EM, Verbeek FJ, Aten JA. Measurement of co-localization of objects in dual-colour confocal images. J Microsc Oxford 1993;169:375–82.
[5] Ramírez O, García A, Rojas R, Couve A, Härtel S. Confined displacement algorithm determines true and random colocalization in fluorescence microscopy. J Microsc 2010;239:173–83.
[6] Dunn KW, Kamocka MM, McDonald JH. A practical guide to evaluating colocalization in biological microscopy. Am J Physiol Cell Physiol 2011;300:C723–42.
[7] Marx V. Is super-resolution microscopy right for you? Nat Methods 2013;10:1157–63.
[8] Sekar RB, Periasamy A. Fluorescence resonance energy transfer (FRET) microscopy imaging of live cell protei localizations. J Cell Biol 2003;160:629–33.
[9] Martin WN, Aggarwal JK. Volumetric descriptions of objects from multiple views. IEEE Trans Pattern Anal Mach Intell 1983;5:150–8.
[10] Kutulakos K, Seitz S. A theory of shape by space carving. In: Proc. IEEE Int. Conf. on computer vision; 1999. p. 307–14.
[11] Yuan X, Zhang N, Nguyen MX, Chen B. Volume cutout. Vis Comput 2005;21:745–54.
[12] Chen H-L J, Samavati FF, Sousa MC. GPU-based point radiation for interactive volume sculpting and segmentation. Vis Comput 2008;24:689–98.
[13] Owada S, Nielsen F, Igarashi T, Haraguchi R, Nakazawa, K. Projection plane processing for sketch-based volume segmentation. In: Proc. IEEE Int. Symp. on biomedical imaging; 2008. p. 117–20.
[14] Abeysinghe S, Ju T. Interactive skeletonization of intensity volumes. Vis Comput 2009;25:627–35.
[15] Wan Y, Otsuna H, Chien C-B, Hansen C. Interactive extraction of neural structures with user-guided morphological diffusion. In: Proc. IEEE Symp. on biological data visualization; 2012. p. 1–8.
[16] Wan Y, Otsuna H, Kwan KM, Hansen C. Real-time dense nucleus selection from confocal data. In: Proc. eurographics workshop on visual computing for biological and medicine; 2014. p. 59–68.
[17] Lefohn AE, Kniss JM, Hansen CD, Whitaker RT. A streaming narrow-band algorithm: interactive computation and visualization of level sets. IEEE Trans Vis Comput Graphics 2004;10:422–33.
[18] Lundstrm C, Ynnerman A, Ljung P, Persson A, Knutsson H. The alpha-histogram: using spatial coherence to enhance histograms and transfer function design. In: EUROVIS - Eurographics /IEEE VGTC Symposium on visualization; 2006. p. 227–34.
[19] Hadwiger M, Laura F, Rezk-Salama C, Höllt T, Geier G, Pabel T. Interactive volume exploration for feature detection and quantification in industrial CT data. IEEE Trans Vis Comput Graphics 2008;14:1507–14.
[20] Lindholm S, Ljung P, Lundström C, Persson A, Ynnerman A. Spatial conditioning of transfer functions using local material distributions. IEEE Trans Vis Comput Graphics 2010;16:1301–10.
[21] Wan Y, Otsuna H, Hansen C. Synthetic brainbows. Comput Graph Forum 2013;32:471–80.
[22] Livet J, Weissman TA, Kang H, Draft RW, Lu J, Bennis RA, Sanes JR, Lichtman JW. Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system. Nature 2007;450:56–62.
[23] von Neumann J, Burks AW. Theory of self-reproducing automata. Urbana and London: University of Illinois Press; 1966.
[24] Chazan D, Miranker W. Chaotic relaxation. Linear Algebra Appl 1969;2:199–222.
[25] Schneider CA, Rasband WS, Eliceiri KW. NIH image to imagej: 25 years of image analysis. Nat Methods 2012;9:671–5.
[26] Rueden CT, Schindelin J, Hiner MC, DeZonia BE, Walter AE, Arena ET, Eliceiri KW. Imagej2: Imagej for the next generation of scientific image data. BMC Bioinf 2017;18:529.
[27] Thermo Fisher Scientific, Inc.. Amira for advanced image processing and quantification. https://www.thermofisher.com/us/en/home/industrial/electron-microscopy/electron-microscopy-instruments-workflow-solutions/3d-visualization-analysis-software.html; 2020.
[28] Oxford Instruments, Plc. Imaris. https://imaris.oxinst.com/; 2020.
[29] Gobbetti E, Marton F, Guitián JAI. A single-pass GPU ray casting framework for interactive out-of-core rendering of massive volumetric datasets. Vis Comput 2008;24(7-9):797–806.
[30] Crassin C, Neyret F, Lefebvre S, Eisemann E. GigaVoxels: ray-guided streaming for efficient and detailed voxel rendering. In: Proc. ACM SIGGRAPH symposium on interactive 3D graphics and games (I3D); 2009. p. 15–22.
[31] Fogal T, Schiewe A, Krüger J. An analysis of scalable GPU-based ray-guided volume rendering. In: Proc. IEEE symposium on large-scale data analysis and visualization (LDAV); 2013. p. 43–51.
[32] Meyer J, Borg R, Takanashi I, Lum EB, Hamann B. Segmentation and texture-based hierarchical rendering techniques for large-scale real-color biomedical image data. In: Data visualization, the state of the art; 2003. p. 169–82.
[33] Agus M, Bettio F, Giachetti A, Gobbetti E, Guitián JAI, Marton F, Nilsson J, Pintore G. An interactive 3D medical visualization system based on a light field display. Vis Comput 2009;9:883–93.
[34] Jeong W-K, Schneider J, Turney SG, Faulkner-Jones BE, Meyer D, Westermann R, Reid RC, Lichtman J, Pfister H. Interactive histology of large-scale biomedical image stacks. IEEE Trans Vis Comput Graphics 2010;16:1386–95.
[35] Jeong W, Beyer J, Hadwiger M, Blue R, Law C, Vzquez-Reina A, Reid RC, Lichtman J, Pfister H. SSECRETT and NeuroTrace: interactive visualization and analysis tools for large-scale neuroscience data sets. IEEE Comput Graph 2010;30:58–70.
[36] Fogal T, Krüger J. Tuvok, an architecture for large scale volume rendering. In: Proc. international workshop on vision, modeling, and visualization; 2010. p. 57–66.
[37] Hadwiger M, Beyer J, Jeong W-K, Pfister H. Interactive volume exploration of petascale microscopy data streams using a visualization-driven virtual memory approach. IEEE Trans Vis Comput Graphics 2012;18:2285–94.
[38] Solteszova V, Birkeland A, Stoppel S, Viola I, Bruckner S. Output-sensitive filtering of streaming volume data. Comput Graph Forum 2017;36:249-262.
[39] Holman HA, Wan Y, Rabbitt RD. Developmental GAD2 expression reveals progenitor-like cells with calcium waves in mammalian crista ampullaris. iScience 2020;23:101407.
[40] Kriegel H, Kröger P, Sander J, Zimek A. Density-based clustering. WIRES Data Min Knowl 2011;1:231–40.
[41] Pizer SM, Amburn EP, Austin JD, Cromartie R, Geselowitz A, Greer T, Romeny BH, Zimmerman JB, Zuiderveld K. Adaptive histogram equalization and its variations. Comput Vis Graph Image Process 1987;39:355–68.
[42] Wan Y, Otsuna H, Holman HA, Bagley B, Ito M, Lewis AK, Colasanto M, Kardon G, Ito K, Hansen C. FluoRender: joint freehand segmentation and visualization for many-channel fluorescence data analysis. BMC Bioinf 2017;18:280.
[43] Siek JG, Lee L-Q, Lumsdaine A. The boost graph library: user guide and reference manual. Upper Saddle River: Addison-Wesley Professional; 2001.
[44] Jolliffe IT. Principal component analysis. New York: Springer; 2002.
[45] Holman HA, Poppi LA, Frerck M, Rabbitt RD. Spontaneous and acetylcholine evoked calcium transients in the developing mouse utricle. Front Cell Neurosci 2019;13:186.
[46] Wan Y, Otsuna H, Chien C, Hansen C. An interactive visualization tool for multi-channel confocal microscopy data in neurobiology research. IEEE Trans Vis Comput Graphics 2009;15:1489–96.
[47] Wan Y, Otsuna H, Kwan KM, Hansen C. FluoRender: an application of 2D image space methods for 3D and 4D confocal microscopy data visualization in neurobiology research. In: Proc. IEEE pacific symposium on visualization; 2012. p. 201–8.
[48] Hadwiger M, Sicat R, Beyer J, Krüger J, Möller T. Sparse PDF maps for non-linear multi-resolution image operations. ACM Trans Graph 2012;31:133:1–133:12.
[49] Sicat R, Krüger J, Möller T, Hadwiger M. Sparse PDF volumes for consistent multi-resolution volume rendering. IEEE Trans Vis Comput Graphics 2014;20:2417–26.
[50] Saad D. On-line learning in neural networks. Cambridge: Cambridge University Press; 1998.