networks, coexpression to rescue RNA interference– or CRISPR-CAS9–induced reduction of endogenous transcripts, and expression of ORFs carrying a mutation of interest to allow measurement of the mutation effect in the absence of the wild-type background.

High-level gene coverage, combined with the versatility of Gateway cloning, and full access to OC clones make this collection a unique and valuable resource for the scientific community that should aid in the functional characterization of new protein targets and testing of disease-relevant mutations on a large scale. The OC resource will continue to be expanded in the future to increase human gene coverage, provide additional isoforms where available, provide clones with medically relevant mutations and add additional species, including ORFs from *Xenopus* and *Drosophila*.

Note: Any Supplementary Information and Source Data files are available in the online version of the paper (http://dx.doi.org/10.1038/nmeth.3776).

ACKNOWLEDGMENTS

The authors acknowledge valuable encouragement for initiating the OC from F. Collins (US National Institutes of Health) and E. Harlow (Harvard Medical School). Some of the cDNAs used as PCR templates for ORF cloning and other support were received from the German cDNA Consortium: K. Köhrer (University Düsseldorf, Germany), W. Ansorge (EMBL Heidelberg, Germany), H. Blöcker (Helmholtz Center Braunschweig, Germany), W. Mewes, C. Amid (Helmholtz Center Munich, Germany), J. Lauber, A. Bahr (Qiagen, Hilden, Germany), D. Heubner, R. Wambutt (Agowa, Berlin, Germany), B. Ottenwälder, B. Obermaier (Medigenomix, Ebersberg, Germany), H. Blum, H. Domdey (University Munich, Germany), I. Schupp, S. Bechtel and A. Poustka (DKFZ, Heidelberg, Germany). The German cDNA Consortium was funded by the Federal Ministry of Education and Research (BMBF) in the frame of the German Genome Project (DHGP) and the German National Genome Research Network (NGFN) programs (to S.W.). This work was supported by the Ellison Foundation (grant to M.V. and D.E.H.); the DFCI Institute (Sponsored Research funds to M.V. and D.E.H.); the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan (research grants to Y.H. at the RIKEN Omics Science Center and to P.C. at the RIKEN Center for Life Science Technologies); and the MEXT Genome Network Project (grant to Y.H.).

AUTHOR CONTRIBUTIONS

M.V., D.S.G., J.L., G.T. and D.E.H. founded the OC and conceived of the project. S.W., Y. Hu, G.B., P.C., I.D., Y. Hayashizaki, J.K., O.O., A.R., K.S.-A., J.S., R.W., M.V. and D.E.H. generated and contributed entry clones. S.W., Y. Hu, G.B., P.C., I.D., Y. Hayashizaki, S.H., B.K., J.K., C.K., S.L., A.R., K.S.-A., J.S., R.W., S.Y., M.V., J.L. and D.E.H. generated sequence-verified ORF clones. S.W., Y. Hu, G.B., I.D., T.H., O.H., S.H., A.H.-W., W.J., A.J., C.K., A.L., S.L., J.P., B.S., L.W. and S.Y. performed bioinformatics analysis and clone annotation. C.P., A.A., C.K., A.L., S.L., A.v.B.S. and S.Y. re-arrayed libraries and carried out quality control of ORF clones. C.P., P.H., M.H., A.A., M.B., J.W.H., B.K., C.L., S.L., J.M., T.M., O.O., J.P., A.R., C.S., B.S., A.v.B.S., T.W. and S.Y. performed archiving and distribution of the ORF clone resource to the public. S.W., C.P., P.H., M.H., M.D., O.H., S.L., J.P., C.S. and S.Y. carried out website and database development. S.W., M.H., M.V., D.S.G., J.L., G.T. and D.E.H. led, oversaw and steered development of the consortium and wrote the paper.

COMPETING FINANCIAL INTERESTS

The authors declare competing financial interests: details are available in the online version of the paper (http://dx.doi.org/10.1038/nmeth.3776).

The ORFeome Collaboration: Stefan Wiemann[1,2], Christa Pennacchio[3], Yanhui Hu[4], Preston Hunter[5], Matthias Harbers[6,7], Alexandra Amiet[8], Graeme Bethel[9], Melanie Busse[10], Piero Carninci[7], Mark Diekhans[11], Ian Dunham[9], Tong Hao[12–14], J Wade Harper[15], Yoshihide Hayashizaki[16], Oliver Heil[2], Steffen Hennig[17], Agnes Hotz-Wagenblatt[2], Wonhee Jang[18], Anika Jöcker[1], Jun Kawai[16], Christoph Koenig[17], Bernhard Korn[19], Cristen Lambert[20], Anita LeBeau[21], Sun Lu[22,23], Johannes Maurer[17], Troy Moore[24], Osamu Ohara[25], Jin Park[5], Andreas Rolfs[26], Kourosh Salehi-Ashtiani[12–14], Catherine Seiler[5], Blake Simmons[21,24], Anja van Brabant Smith[8], Jason Steel[5], Lukas Wagner[18], Tom Weaver[10], Ruth Wellenreuther[1], Shuwei Yang[22], Marc Vidal[12–14], Daniela S Gerhard[27], Joshua LaBaer[5,28], Gary Temple[20] & David E Hill[12–14]

[1]Division of Molecular Genome Analysis, German Cancer Research Center (DKFZ), Heidelberg, Germany. [2]Genomics & Proteomics Core Facility, German Cancer Research Center (DKFZ), Heidelberg, Germany. [3]IMAGE Consortium, Lawrence Livermore National Laboratories, Livermore, California, USA. [4]Department of Genetics, Harvard Medical School, Boston, Massachusetts, USA. [5]Virginia G. Piper Center for Personalized Diagnostics (VGPCPD), Biodesign Institute, Arizona State University, Tempe, Arizona, USA. [6]DNAFORM Inc., Tsurumi-ku, Yokohama City, Kanagawa, Japan. [7]Division of Genomic Technologies, RIKEN Center for Life Science Technologies, RIKEN Yokohama Institute, Tsurumi-ku, Yokohama, Kanagawa, Japan. [8]Dharmacon, GE Healthcare, Lafayette, Colorado, USA. [9]Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Cambridge, UK. [10]Source BioScience, Nottingham, UK. [11]UC Santa Cruz Genomics Institute, University of California, Santa Cruz, California, USA. [12]Center for Cancer Systems Biology (CCSB), Dana-Farber Cancer Institute, Boston, Massachusetts, USA. [13]Department of Cancer Biology, Dana-Farber Cancer Institute, Boston, Massachusetts, USA. [14]Department of Genetics, Harvard Medical School, Boston, Massachusetts, USA. [15]Dana-Farber–Harvard Cancer Center (DFHCC) DNA Resource Core and Department of Cell Biology, Harvard Medical School, Boston, Massachusetts, USA. [16]RIKEN Preventive Medicine & Diagnosis Innovation Program, RIKEN Yokohama Institute, Wako, Saitama, Japan. [17]imaGenes GmbH, Berlin, Germany. [18]National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, Maryland, USA. [19]Ressourcenzentrum für Genomforschung gGmbH, Berlin, Germany. [20]National Human Genome Research Institute, National Institutes of Health, Bethesda, Maryland, USA. [21]HudsonAlpha Institute of Biotechnology, Huntsville, Alabama, USA. [22]GeneCopoeia, Inc., Rockville, Maryland, USA. [23]Guangzhou FulenGen, Ltd., Guangdong, China. [24]Open Biosystems, Inc., Huntsville, Alabama, USA. [25]Kasusa DNA Research Institute, Kisarazu, Chiba, Japan. [26]Department of Biological Chemistry & Molecular Pharmacology, Harvard Institute of Proteomics, Harvard Medical School, Boston, Massachusetts, USA. [27]Office of Cancer Genomics, National Cancer Institute, National Institutes of Health, Bethesda, Maryland, USA. [28]Department of Chemistry and Biochemistry, Arizona State University, Tempe, Arizona, USA. Correspondence should be addressed to D.E.H. (david_hill@dfci.harvard.edu), S.W. (s.wiemann@dkfz.de), G.T. (gftemple@gmail.com), M.H. (matthias.harbers@riken.jp), M.V. (marc_vidal@dfci.harvard.edu) or J.L. (joshua.labaer@asu.edu).

1. Walhout, A.J. *et al. Methods Enzymol.* **328**, 575–592 (2000).
2. MGC Project Team *et al. Genome Res.* **19**, 2324–2333 (2009).
3. Wiemann, S. *et al. Genome Res.* **11**, 422–435 (2001).
4. Yang, X. *et al. Nat. Methods* **8**, 659–661 (2011).
5. Rolland, T. *et al. Cell* **159**, 1212–1226 (2014).
6. Yu, X. & LaBaer, J. *Nat. Protoc.* **10**, 756–767 (2015).

TeraFly: real-time three-dimensional visualization and annotation of terabytes of multidimensional volumetric images

To the Editor: New sample preparation and high-throughput light-sheet microscopy techniques[1] are increasingly capable of generating multidimensional (3D and higher) images easily exceeding the terabyte size. This has posed a significant challenge for scalable interactive visualization and quantitative annotation of such big image data. A common practice is to design a data-streaming and visualization tool to supply and display small parts of an image volume when needed[2,3]. However, existing tools allow only 2D slice-based rendering of 3D image stacks. Such 2D approaches not only are time consuming and low throughput but also bring bias to the understanding of intrinsic 3D properties of bioimage data[4]. A free, open-source and cross-platform software tool for true 3D visualization and 3D annotation of very large multidimensional volumes is highly desired (**Supplementary Note 1**).

To fill this gap, we have developed TeraFly software for interactive 3D visualization of terabytes of 3D and 4D (3D spatial information plus color) images, as well as 5D (4D plus time) image series, with subsecond response times from both local and remote data sources (**Supplementary Note 2**). TeraFly instantly translates simple computer-mouse actions (e.g., drags and scrolls) performed directly in the volumetric space of a 3D viewer into translation, rotation and zoom for the 3D volumes of interest (VOIs) displayed at the appropriate image resolution (**Supplementary Videos 1** and **2**). TeraFly uses only 336 megabytes of computer memory to display a 1-teravoxel image stack with three color channels (i.e., $10{,}000^3$ voxels) and would use only 480 megabytes for a 1-petavoxel, three-channel image stack (i.e., $100{,}000^3$ voxels) (**Supplementary Note 2**).

Although TeraFly adopts an often-used multiresolution pyramid image organization for fast data accessing (**Fig. 1a**) and is able to generate such hierarchical data much more efficiently than BigDataViewer[2] (2–7× the speed and 30–74× the memory savings) (**Supplementary Note 3**), we believe that the optimization of data organization alone is insufficient to achieve a real-time response when a true 3D rendering (e.g., real-time maximum intensity projection or alpha blending) is considered. Thus, we implemented two critical techniques to boost TeraFly (**Supplementary Note 4**). The first is a 'mean shift of mean shift' method to accurately and instantly (typically <10 ms) estimate the 3D VOI when the user is zooming in on the displayed image content (**Fig. 1b**). This solves the challenging problem of mapping the 2D user input (i.e., the mouse position on the screen) to the 3D VOI the user sees at a higher magnification level. The second is an effective fetch-and-display strategy to instantly show an interpolated low-resolution version of the VOI's image t while the higher-resolution data are quickly loaded and filled in.

TeraFly enables quantitative analysis of big image data with minimal human effort. The annotation of displayed biological structures is done directly in the volumetric space of a 3D image stack (hence the term '3D annotation') with simple computer-mouse gestures such as a single click or a single stroke. This includes the generation and curation of various surface objects such as 3D markers and tubes that are used, for instance, for cell counting or the tracing of long neurites (e.g., in a whole-brain image) or, further, to generate a gold standard to feed semi- or fully supervised image analysis algorithms, as well as for proofreading or evaluating the output of these algorithms (**Supplementary Note 5**). We also designed a dedicated image-exploration modality to boost users' proofreading performance (**Supplementary Video 3**) and used it to generate the most complete and precise map of Purkinje cells in a whole mouse cerebellum ever obtained[5] (**Fig. 1c** and **Supplementary Note 6**). Compared with other tools based on 2D annotation, our approach was considerably faster and more precise (**Supplementary Videos 4** and **5** and

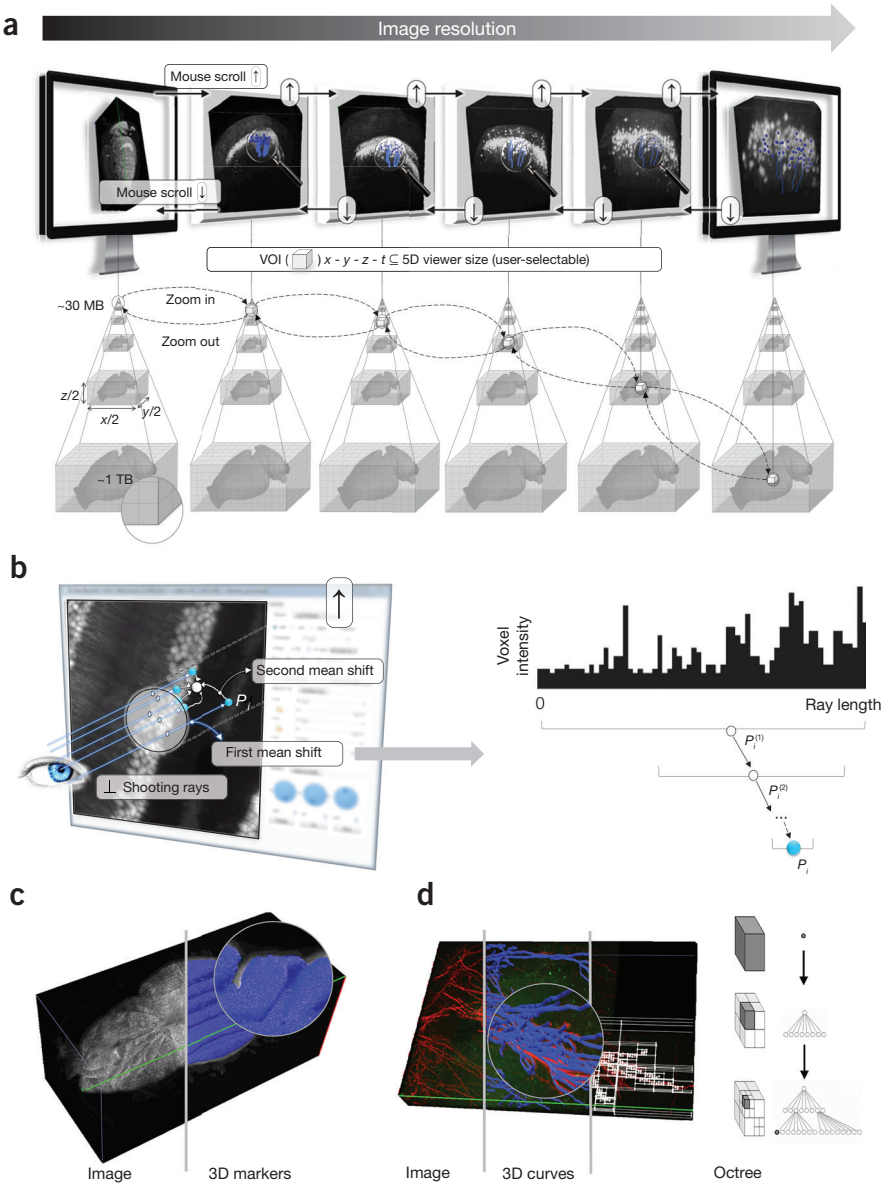


Figure 1 | Overview of TeraFly image and surface-object visualization. (**a**) 3D image exploration based on progressively higher-resolution VOIs fetched from a multiresolution tiled image pyramid data structure and displayed in distinct 3D viewers (one per pyramid layer) synchronized for zooming in and zooming out. In this example, an entire mouse brain image about 1 terabyte (TB) in size was recursively downsampled five times until the entire image could fit into the 3D viewer at the lowest resolution. (**b**) Schematic illustration of the 3D VOI estimation–based 'mean shift of mean shift'. First, the mean-shift method is applied on the voxel intensity along each of the shooting rays for the calculation of the corresponding bio-entity 3D locations. Then the mean-shift strategy is again applied on these 3D locations for the calculation of the VOI center. (**c**) Image of L7-GFP whole mouse cerebellum (110.1 gigabytes) with an overlay displaying 220,800 TeraFly-curated Purkinje cells as 3D markers. (**d**) Image of adult rat neuron acquired with two-photon microscopy with overlaid TeraFly-assisted 3D tracings generated at different resolution scales and the corresponding octree-based 3D curve point representation.

Supplementary Note 6). TeraFly also allows us to perform efficient 3D annotation for complicated biological structures (e.g., rat, mouse and human neurons) in very large multidimensional images (**Fig. 1d**, **Supplementary Video 6** and **Supplementary Note 6**).

We implemented these 3D annotation functionalities by leveraging the built-in 'Virtual Finger' algorithms of Vaa3D[6], which map users' inputs in the 2D plane of a computer screen to the 3D locations of the corresponding biological structures. However, this alone would not allow the efficient handling of the millions of 3D object points (e.g., marker centers and curve nodes) that are likely to be produced by the computerized analysis of big images. Thus, we used an octree data structure to encode the annotated (or automatically produced) 3D objects (**Fig. 1d**). We generated a lookup table for efficient representation, search and resampling of such 3D annotation data with respect to any VOI (**Supplementary Note 7**).

We have applied TeraFly to several huge image data sets from different modalities at the Allen Institute for Brain Science, Janelia Research Campus of the Howard Hughes Medical Institute, the European Human Brain Project and other places. To our knowledge, TeraFly is the first free, open-source, cross-platform software tool for 3D integrated visualization and annotation of massive image data.

TeraFly has been implemented in C++ and is included in the default Vaa3D installation available at http://www.vaa3d.org/.

Note: Any Supplementary Information and Source Data files are available in the online version of the paper (http://dx.doi.org/10.1038/nmeth.3767).

ACKNOWLEDGMENTS

We thank F.S. Pavone, L. Sacconi, L. Silvestri, J.P. Ghobril, R. Tsien, H. Zeng, P. Keller and E. Betzig for providing the data sets we used in this work, as well as for the useful discussions that helped us in the requirement analysis of our tool.

AUTHOR CONTRIBUTIONS

H.P. conceived the project while collaborating with A.B. and G.I. A.B., G.I. and H.P. developed and tested the software and wrote the paper. L.O. proofread automatic cell counts and tested the software.

COMPETING FINANCIAL INTERESTS

The authors declare no competing financial interests.

Alessandro Bria[1–3], Giulio Iannello[1], Leonardo Onofri[1] & Hanchuan Peng[3]

[1]Department of Engineering, University Campus Bio-Medico of Rome, Rome, Italy. [2]Department of Electrical and Information Engineering, University of Cassino and Southern Lazio, Cassino, Italy. [3]Allen Institute for Brain Science, Seattle, Washington, USA. Correspondence should be addressed to H.P. (hanchuanp@alleninstituite.org).

1. Tomer, R., Ye, L., Hsueh, B. & Deisseroth, K. *Nat. Protoc.* **9**, 1682–1697 (2014).
2. Pietzsch, T., Saalfeld, S., Preibisch, S. & Tomancak, P. *Nat. Methods* **12**, 481–483 (2015).
3. Peng, H., Ruan, Z., Long, F., Simpson, J.H. & Myers, E.W. *Nat. Biotechnol.* **28**, 348–353 (2010).
4. Long, F., Zhou, J. & Peng, H. *PLoS Comput. Biol.* **8**, e1002519 (2012).
5. Silvestri, L. *et al. Front. Neuroanat.* **9**, 68 (2015).
6. Peng, H. *et al. Nat. Commun.* **5**, 4342 (2014).

Supplementary Note 1. Related software

Driven by continuous advances in microscopy and related technologies, bioimage informatics that tackles problems such as the computational analysis of biological images is becoming more and more important (Peng, 2008; Walter, et al., 2010; Long, et al., 2012). Over the last 20 years, a number of tools for visualizing, annotating, and quantitatively analyzing multidimensional biological image data have been developed. They include public-domain tools such as ScanImage (Pologruto, et al., 2003), µManager (Stuurman, et al., 2010), MicroPilot (Conrad, et al., 2011), ImageJ (Abrmoff, et al., 2004), Vaa3D (Peng, et al., 2010), Ilastik (Sommer, et al., 2011), CellProfiler (Carpenter, et al., 2006), CellOrganizer (Murphy, 2012), CellExplorer (Long, et al., 2009), BrainExplorer (Lau, et al., 2008), ClearVolume (Royer, et al., 2015), and many commercial software suites such as Zen (Zeiss), Amira (VSG), Imaris (Bitplane), ImagePro (MediaCybernetics), Neurolucida (MBF Bioscience).

The major limitation of previous tools is that when applied to the terabyte-size images currently generated by modern microscopy techniques (Silvestri, et al., 2012; Tomer, et al., 2014) it is hard to load the entire image volume into computer memory quickly, let alone the fact that most current computers do not even have enough memory to hold such big image data. Consequently, scalable solutions that use multiresolution approaches have been recently proposed (Saalfeld, et al., 2009; Jeong, et al., 2010; Pietzsch, et al., 2015; Amat, et al., 2015; Tinevez & Pietzsch, 2015). They assume that image data are stored in a hierarchical data structure where each hierarchical level represents a different resolution that can be independently loaded upon requests. Moreover, the data at each resolution are stored in such a way that only the data corresponding to the Region of Interest (ROI) have to be loaded into main memory for visualization or processing. Being able to select the resolution and the ROI makes it possible to deal with datasets exceeding the available resources (Peng, et al., 2014).

CATMAID (Saalfeld, et al., 2009) allows rapid, uninterrupted browsing of multi-terabyte data sets and concurrent large-scale data annotation involving tens of millions of data points, even when accessing the data remotely through the internet. CATMAID was initially developed for visualizing and annotating large electron microscopy data sets generated in the field of connectomics, but it was recently extended to support large light microscopy image data sets with up to five dimensions (three spatial dimensions, color and time) (Amat, et al., 2015). CATMAID and its branches are accessible through an internet browser and display image data superimposed with cell-lineage data points in a tri-view arrangement (XY, YZ and XZ slices of the specimen). SSECRETT (Jeong, et al., 2010) is a 2D slice-based volume exploration and manual annotation tool for extremely large-scale neuroscience datasets. It is based on a client-server architecture where the dataset resides on the server side and the client can request an arbitrary 2D cross-section view of the dataset. BigDataViewer (Pietzsch, et al., 2015) is a Fiji (Pietzsch, et al., 2012) plugin to interactively navigate and visualize virtual 2D slices from very large 5D terabyte-sized images from both local and remote data sources. The tool is based on a custom HDF5 based data format that is optimized for fast arbitrary re-slicing of the image at various scales. HDF5 (The HDF Group, 2014), the last version of the Hierarchical Data Format is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for

high volume and complex data. MaMuT (Tinevez & Pietzsch, 2015) is a Fiji plugin that provides interactive visualization, annotation, tracking and lineaging of very large, multiview image datasets. MaMuT builds on TrackMate (Pietzsch, et al., 2012), a Fiji plug-in for single particle tracking, and uses BigDataViewer both as a data backend and as a visualization frontend.

Unfortunately, the tools aforementioned are all based on 2D cross-sectional views or a combined display with an arbitrary-angle cutting plane, which is often insufficient to observe complex 3D structures and the relationship among multiple objects (e.g., cells) in a 3D or higher-dimensional image. This leads to the inability to efficiently explore the complicated 3D image content and thus to input user-specified information of the observed image patterns directly in the 3D space (Long, et al., 2012). Hence, it is desirable to have efficient tools that integrate 3D visualization and annotation functions but that can scale well on very large (terabyte-sized) images. Although a few scalable tools provide preliminary capabilities of 3D visualization and annotation modules in the context of very large scale of image datasets, these tools are limited owing to high expense of licenses (Arivis (AG), Amira (VSG), Imaris (Bitplane)) and infrastructures (Paraview (Kitware Inc.)). This presents an obstacle for the unbiased, high-throughput and quantitative analysis of bioimage data and creates tremendous need for the development of new techniques that help explore very large 3D data directly and efficiently without expensive virtual reality devices and/or parallel computing infrastructures.

To our knowledge, TeraFly is the first free, open-source, and cross-platform software tool for true 3D visualization and 3D annotation of very large multidimensional volumes. In Table 1.1, we compare TeraFly with the best available tools in the field. We limit our comparison to software that (i) has been reported to visualize multidimensional (up to 5D) data sets whose size exceeds 1 terabyte on commonly available hardware (therefore Table 1.1 excludes all the tools mentioned in the first paragraph of this section and the Paraview software (Kitware Inc.)); and (ii) natively supports time series (therefore Table 1.1 excludes SSECRET).

	Arivis (AG)	BigDataViewer (Pietzsch, et al., 2015)	CATMAID (Saalfeld, et al., 2009)	CATMAID (Amat, et al., 2015)	MaMuT (Tinevez & Pietzsch, 2015)	TeraFly
rendering	both 2D and 3D	2D (arbitrary reslicing)	2D (tri-view)	2D (tri-view)	2D (arbitrary reslicing)	both 2D and 3D
annotation approach	2D slice-by-slice	N/A	2D slice-by-slice	2D slice-by-slice	2D slice-by-slice	3D
custom image format	SIS [a]	BDV HDF5	tile scheme [b]	Keller Lab Block	BDV HDF5	hierarchy of TIFFs or Vaa3D raw files; and BDV HDF5
license	proprietary	open	open	open	open	open

Table 1.1. Visualization software for multidimensional multi-terabyte image data. (a) proprietary data format. (b) Data stored as small 2D-tiles representing a 2D-scale pyramid following a primitive naming scheme.

Supplementary Note 2. Visualization of Big-Image-Data

2.1 Introduction

TeraFly builds on top of the free, open-source, cross-platform Vaa3D system and extends its powerful 3/4/5D image rendering capabilities (Peng, et al., 2010) to images of potentially unlimited size. To achieve this goal, similarly to other tools designed for big image data visualization (**Supplementary Note 1**), TeraFly adopts a multiresolution pyramid image that enables fast access of small parts of an image volume at different scales (**Supplementary Note 3**). Such hierarchical data organization alone, however, is insufficient to achieve a real-time response when a true 3D rendering (e.g. real-time maximum intensity projection or alpha blending) is considered. Thus, throughout the 3D exploration of the image, other critical solutions intervene to boost the visualization performance (**Supplementary Notes 4 and 5**). In this supplement, we describe the overall 3D image exploration approach and quantitatively characterize the performance of our tool in terms of memory usage and visualization response time.

2.2 3D image exploration

Once the image is opened in TeraFly, the 3D image exploration starts by loading and displaying the entire image content of the highest (and coarsest) multiresolution pyramid level, which by construction can fit into the 3D viewer (**Supplementary Note 3**). As the user zooms-in with the mouse scroll wheel into this coarse resolution image, TeraFly instantly (<10 ms typically) generates the 3D Volume of Interest (VOI) best approximating the region currently viewed (**Supplementary Note 4**), then loads the higher resolution data corresponding to this VOI and renders it in a new 3D viewer quickly (**Supplementary Note 5**). Further zooms-in are processed in the same way until the lowest level of the pyramid is reached. To zoom-out, TeraFly goes back up through the image pyramid and redisplays the previously viewed VOIs. Leveraging the powerful of Vaa3D's multiple 3D viewers, TeraFly selectively displays and hides the 3D viewers corresponding to different VOIs, thus giving the clue of a smooth 3D exploration of the image (**Supplementary Videos 1-2**).

2.3 Memory usage

The 3D image exploration strategy previously described allows avoiding more complicated data synchronization and caching techniques, which would need to be tailored to the underlying hardware infrastructure. Instead, a simple caching strategy is used: throughout the navigation, the displayed 3D viewers are cached in the graphic card memory and quickly restored both when zooming-out, and when zooming-in a previously viewed VOI. Assuming 8 bits per pixel for display, this requires storing in the graphic card memory $(k+1) \times B_x \times B_y \times B_z \times c \times B_t$ bytes at most, being k the number of pyramid layers, B_x, B_y, B_z and B_t the dimensions of the 5D viewer along x, y, z (space) and t (time) axes (default value is $256(x) \times 256(y) \times 256(z) \times 1(z)$), and c the number of channels. Based on the criterion to choose k (**Supplementary Note 3**), and after simple algebraic manipulations, the maximum memory requirement for a cubic image of N pixels is $\left\lceil \left(\log N^{1/3} - \log B\right) \log_2 e + 1 \right\rceil \times B^3 \times c$ bytes where we chose $B_x = B_y = B_z = B$ and $B_t = 1$ (i.e.

one time-point loaded and displayed at a time). Remarkably, this corresponds to only 336 megabytes of computer memory for an image stack with one teravoxel and three color channels (i.e. $10{,}000^3$ voxels), and to only 480 megabytes for 1 petavoxel three-channels image stack (i.e. $100{,}000^3$ voxels).

2.4 Benchmarks

We tested the visualization of various 3D/4D image volumes of rat neurons and mouse brains with sizes ranging from 0.3 gigabyte to 2.5 terabyte (Table 2.1). The statistics for each test case were obtained by an experienced user on at least 100 trials of randomly selected target VOIs in arbitrarily determined scales of the respective images. The total time for generation, loading and displaying of a 3D VOI is reported in Fig. 2.1a for a MacBook Pro Retina connected to a 16 terabyte QNAP TS-420 Network Attached Storage (NAS) via 1 Gbps LAN network. Similarly, we tested the visualization of four 5D image stacks of zebrafish embryos with sizes ranging from 5 gigabyte to 1.3 terabyte (Table 2.2). Performance times for this experiment are reported in Fig. 2.1b. Remarkably, in all the test cases considered the time scaled constantly on image size and remained always within 1 second, regardless of the overall size and dimensions (3D, 4D, or 5D) of the data sets tested, thus demonstrating that TeraFly can potentially smoothly visualize even larger multi-dimensional image stacks.

Dataset	**Dimensions** ($x \times y \times z \times c$)	**Size** (gigabyte)
Purkinje cells [(a)]	$800 \times 800 \times 512 \times 1$	0.3
Rat neuron [(b)]	$9{,}640 \times 6{,}952 \times 179 \times 2$	24.0
Whole mouse cerebellum [(a)]	$8{,}249 \times 3{,}662 \times 3{,}646 \times 1$	110.1
Mouse hippocampus [(a)]	$9{,}722 \times 8{,}378 \times 5{,}145 \times 1$	419.1
Whole mouse brain [(a)]	$14{,}261 \times 6{,}814 \times 7{,}828 \times 1$	760.7
Whole mouse brain [(c)]	$40{,}000 \times 30{,}000 \times 700 \times 3$	2,520.0

Table 2.1. 3D/4D image volumes used to test the visualization performance. (a) Acquired using Confocal Light Sheet Microscopy *(Silvestri, et al., 2012)* for the Human Brain Project (courtesy of F.S. Pavone, L. Sacconi, L. Silvestri). (b) Acquired using 2-photon microscopy (courtesy of R. Tisen). (c) Acquired using TissueCyte 2-photon imaging system (courtesy of H. Zeng).

Dataset	**Dimensions** ($x \times y \times z \times c \times t$)	**Size** (gigabyte)
Betzig fish data K	$992 \times 794 \times 59 \times 1 \times 100$	4.7
Betzig fish data B1	$992 \times 992 \times 231 \times 2 \times 200$	90.1
Betzig fish data B2	$992 \times 992 \times 231 \times 2 \times 1{,}000$	454.6
Keller	$1{,}556 \times 700 \times 122 \times 2 \times 5{,}000$	1,328.8

Table 2.2. 5D light-sheet microscopy image volumes used to test the visualization performance.

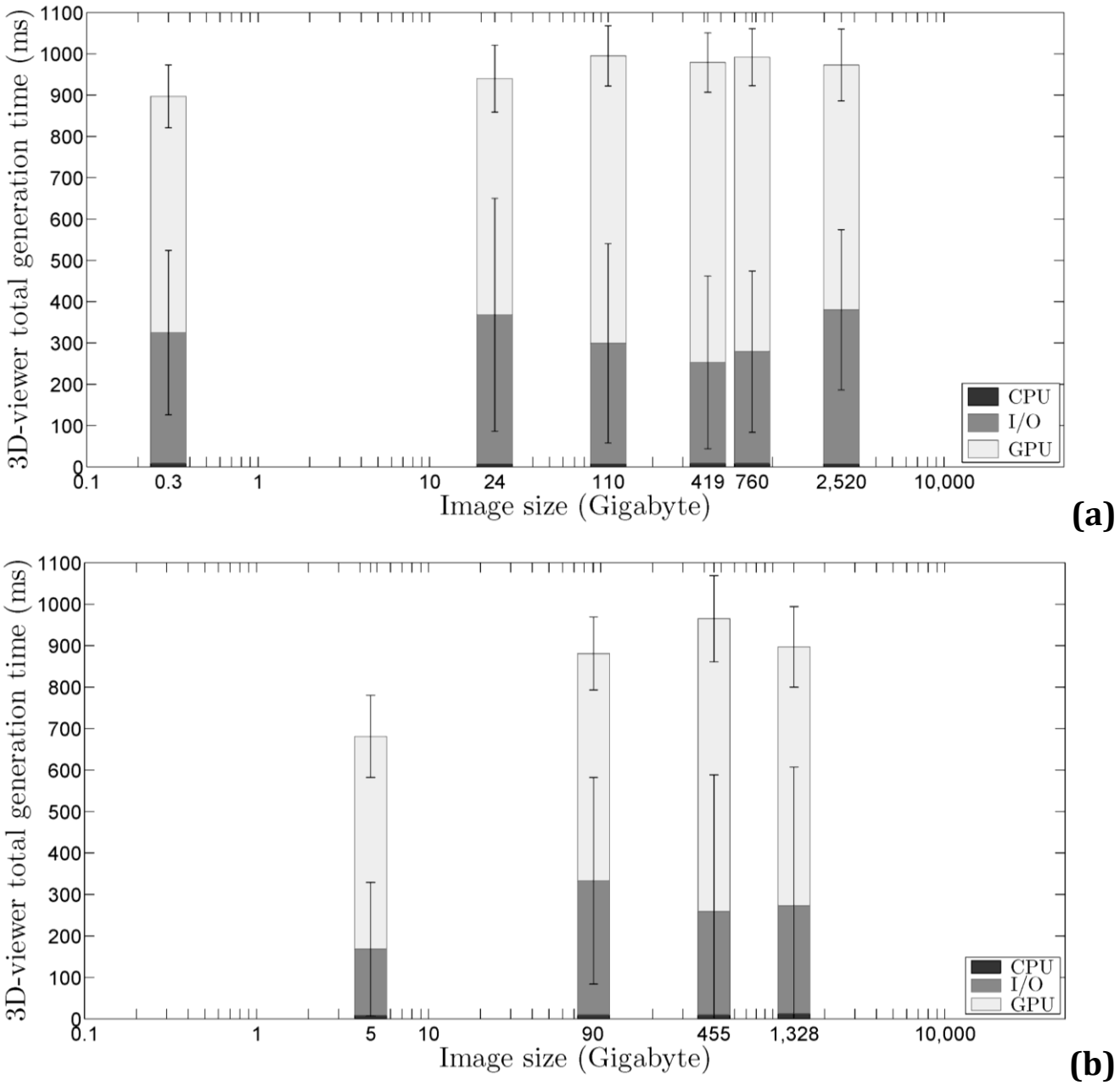


Fig. 2.1. (a) Average total time for generation, loading and displaying of a 3D VOI for six 3D/4D image stacks with size ranging from 0.3 gigabyte to 2.5 terabyte (Table 2.1). (b) Average total time for generation, loading and displaying of a 3D VOI of one time frame at a time on four 5D image stacks with size ranging from 5 gigabyte to 1.3 terabyte (Table 2.2). For each test case, average generation (CPU), loading (I/O) and displaying (GPU) times are reported using different shades of gray with error bars being twice the standard deviation of the data.

Supplementary Note 3. File Format

3.1 Introduction

TeraFly adopts a multiresolution tiled pyramid image format that enables fast access of small parts (tiles) of an image volume at different resolution scales. Section 3.2 gives the specification of this format, Section 3.3 describes how to export a dataset using TeraConverter (included in Vaa3D), and Section 3.4 provides benchmarks results for some of the datasets used in our experiments.

3.2 Format specification

Each TeraFly dataset contains a set of 3D image tiles, stored either as multipage TIFF (Adobe Developers Association, 1992) or Vaa3D raw files (Peng, et al., 2010). The tile files are organized in a hierarchy of nested folders composed by 6 levels (see Fig. 3.1). It is important to note that by making individual tiles available as separate 3D image stacks and in an accessible format (i.e. in a standard 3D TIFF or the Vaa3D raw file format that is used in several very large scale neuroscience projects), we enable very flexible ways to access the data at both global and local scales at any time when a user needs such data, as well as providing more robustness of the stored files in cases of possible damage of the storage media (i.e. hard-drive failure). Our format also makes it possible to save storage-space when the image content is sparse.

The complete specification of our hierarchical data format is given by the 6 levels of the hierarchy, defined as follows:

- ℓ : contains the scale layers of the multiresolution pyramid, each stored into a folder named RES($\text{dim}_y \times \text{dim}_x \times \text{dim}_z$), with dim_y, dim_x, and dim_z being the dimensions (in voxels) of the image along y, x, and z, respectively;
- t : contains the time points stored into folders named T_tttttt, with tttttt being 6 digits (0-9) identifying the coordinate along t (000000 for $t = 0$);
  optional: if there is just one time point, this level can be omitted (see Fig. 3.1, 4D format)
- c : contains the channels stored into folders named CH_cc, with cc being the channel index;
  optional: for grayscale or color (RGB) images, this level can be omitted (see Fig. 3.1, 3D format)
- y : contains the tiles grouped by rows, stored into separate folders named in ascending alphanumeric order. The higher the alphanumeric value, the higher the y coordinate;
  optional: folder names can optionally encode physical coordinates following the yyyyyy convention, where yyyyyy are 6 digits (0-9) identifying the coordinate along y in terms of u space units. For instance, if $u = 0.1\ \mu m$ and yyyyyy = 168270, then the row is associated to the physical coordinate $y = 16.827\ mm$.
- x : contains the tiles grouped by columns, stored into separated folders named in ascending alphanumeric order. The higher the alphanumeric value, the higher the x coordinate;
  optional: folder names can optionally encode physical coordinates following the yyyyyy_xxxxxx convention, where yyyyyy and xxxxxx are 12 digits (0-9) identifying the coordinate y and x, respectively, in terms of u space units.
- z : contains the 3D image tiles stored either as compressed multipage TIFF (.tif) or as uncompressed Vaa3D raw (.raw) files, named in ascending alphanumeric order. The higher

the alphanumeric value, the higher the z coordinate. Each file is a 3D image whose dimensions along x, y, and z are T_x, T_y, and T_z, respectively.

critical: T_y must be the same for all the tiles contained in the current row level y, and it defines the height (in voxels) of that row (all rows sum to $\dim_y$). T_y must be the same for all the tiles contained in the current column level x, and it defines the width (in voxels) of that column (all columns sum to $\dim_x$). T_z can be different for the image files of the current level z, but they must sum to $\dim_z$.

optional: image filenames can optionally encode physical coordinates following the `yyyyyy_xxxxxx_zzzzzz` convention, where `yyyyyy`, `xxxxxx` and `zzzzzz` are 18 digits (0-9) identifying the coordinate y, x, and z, respectively, in terms of u space units.

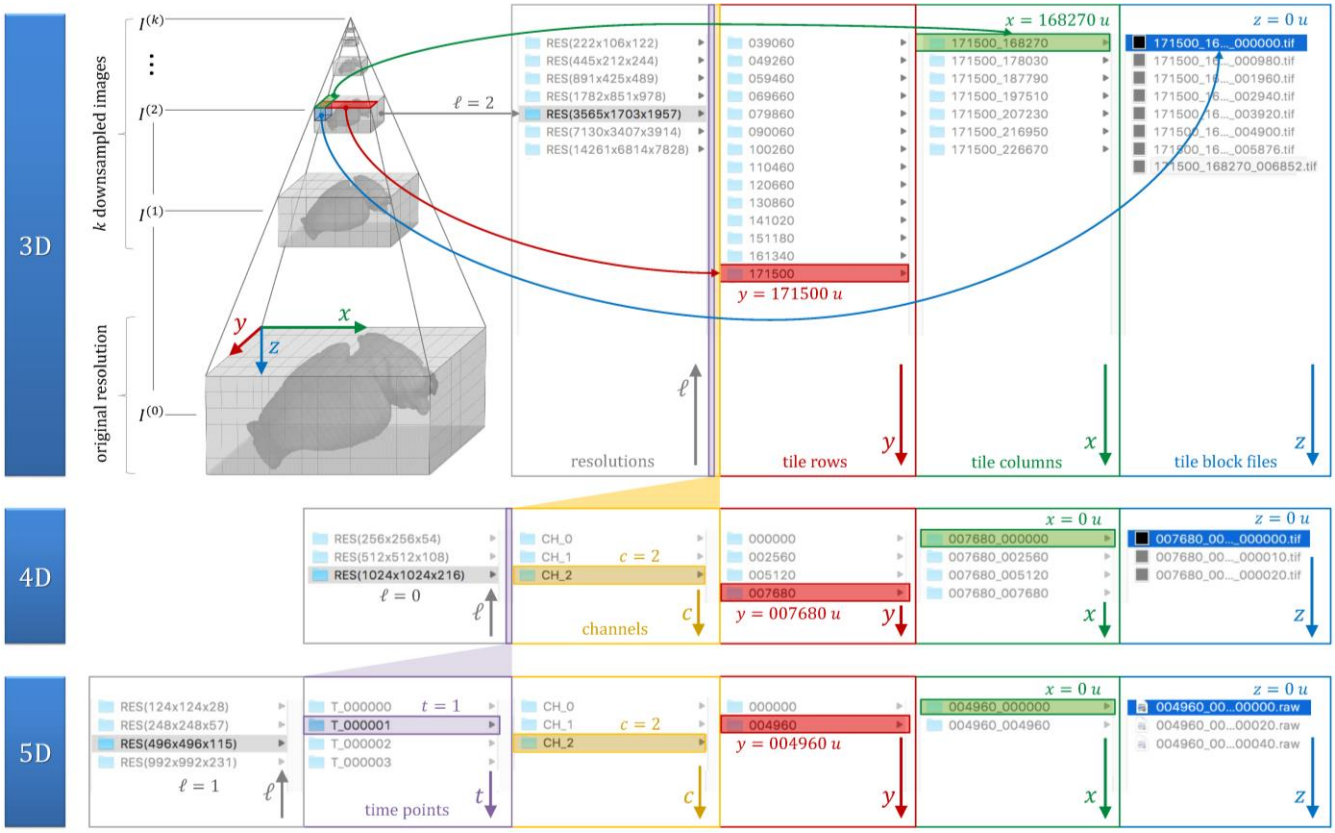


Fig. 3.1. Schema of 3D, 4D, and 5D TeraFly formats. In the 3D example, an entire mouse brain image of size about 1 terabyte has been recursively downsampled $k = 6$ times. In the same example, the tile subdivision at the various levels of the pyramid has only illustrative purposes, and does not reflect the actual number of tiles along x, y and z.

In addition to the 3D image files, each TeraFly dataset also contains the following metadata files:

- vmap.bin: a binary file stored at the level ℓ of the hierarchy of folders and containing the image data that are instantly fetched and displayed when a volume is opened in TeraFly. This file is generated automatically when a volume is opened for the first time, and the image data are fetched from the appropriate resolution scale according to the actual settings of the 3D viewer dimension (the higher the 3D viewer size, the higher the resolution)(see **Supplementary Note 5**, Section 5.3.4(ii));

- cmap.bin: a binary file stored at the level c of the hierarchy of folders and containing internal TeraFly metadata allowing efficient access to the image channels. This file is generated automatically when a dataset is exported for TeraFly with TeraConverter. If this file is missing, TeraFly can automatically regenerate it upon request (see **Supplementary Note 5**, Section 5.3.1);
- mdata.bin: a binary file stored at the level y of the hierarchy of folders and containing internal TeraFly metadata allowing efficient access to the 3D image tiles. This file is generated automatically when a dataset is exported for TeraFly with TeraConverter; If this file is missing, TeraFly can automatically regenerate it upon request (see **Supplementary Note 5**, Section 5.3.1).

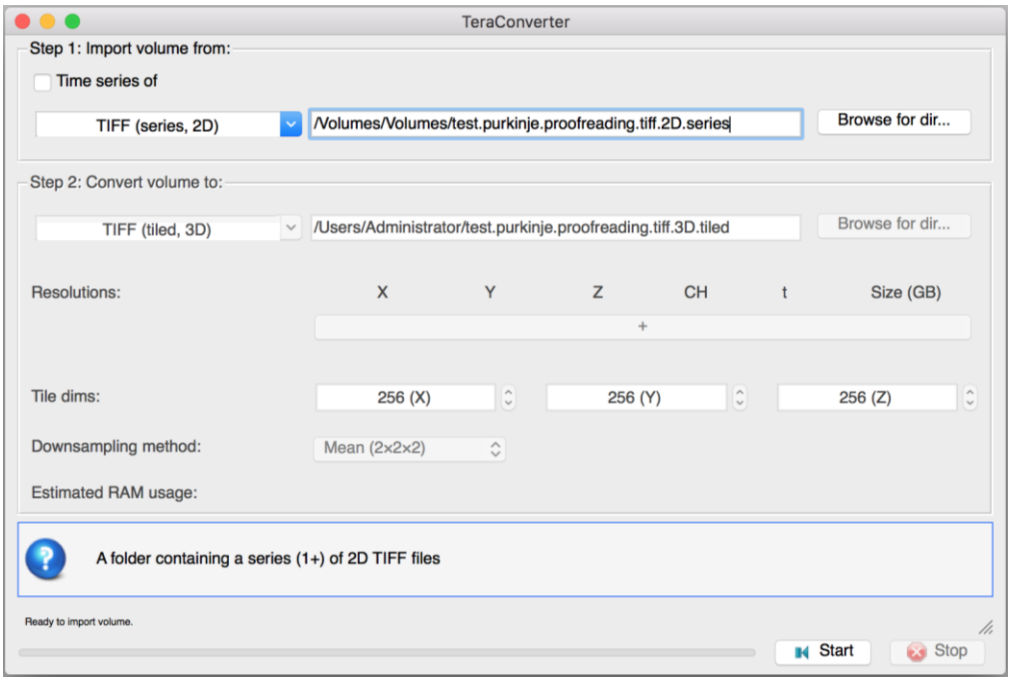
3.3 Exporting data sets for TeraFly with TeraConverter

Based on the underlying graphic hardware capabilities, TeraConverter generates the pyramid image as follows. Let be I the original, highest-resolution, very large-sized image to be imported. Starting from I, another k images $\{I^{(1)}, I^{(2)}, \ldots, I^{(k)}\}$ are obtained by recursively downsampling by two $I^{(j)}$ from $I^{(j-1)}$ $\forall j = 1, \ldots, k$, where $I^{(0)} = I$. This process is iterated until the lowest-resolution image $I^{(k)}$ fits within the 3D viewer of size $B_x \times B_y \times B_z$. These define the maximum size in voxels of displayable image data in the 3D renderer and can be set by the user from TeraFly (default is $256(x) \times 256(y) \times 256(z)$ voxels) (see **Supplementary Note 5**, Section 5.3.4(ii)).

To get started, let's open TeraConverter from Vaa3D by

Vaa3D Menu > Advanced > Big-Image-Data > TeraConverter

which brings up the following dialog.



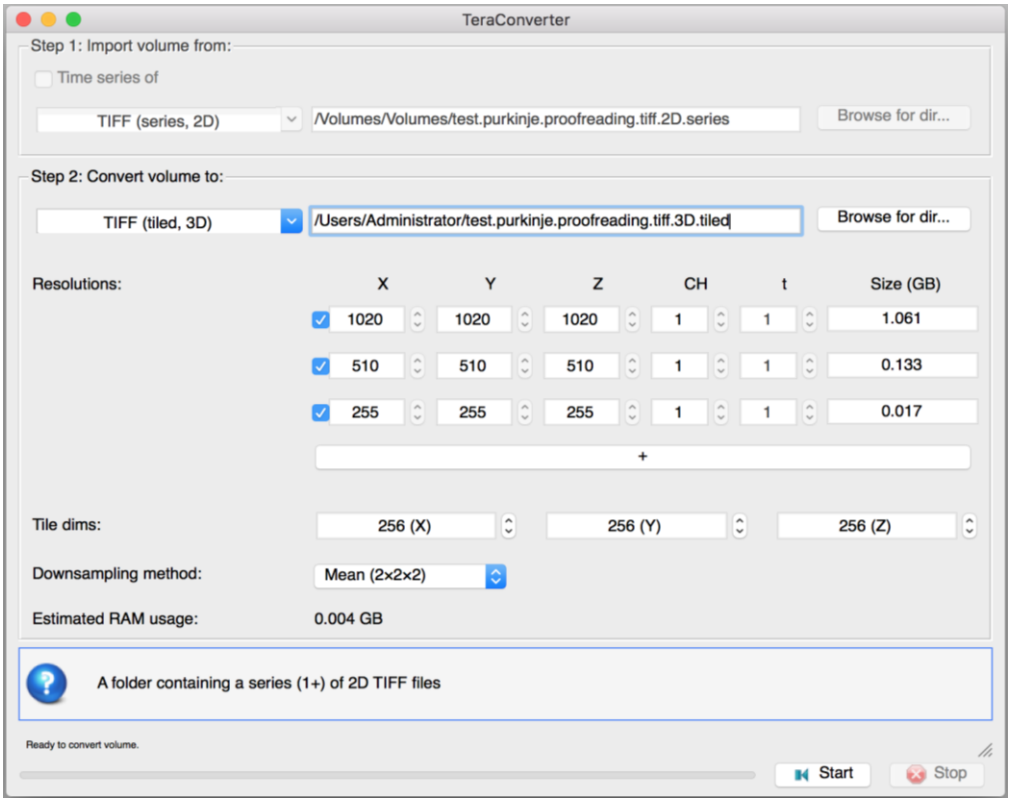
The dialog is divided into four sections (from top to bottom): (i) input form; (ii) output/conversion form; (iii) help box; and (iv) status bar with start/stop button.

In the input form, the user should first specify the input image format from the drop-down menu and then the path where the image files are stored. Supported input formats are:

- TIFF (series, 2D): a folder containing a series (1+) of 2D TIFF files;
- TIFF (3D): single multipage 3D TIFF file;
- TIFF (tiled, 2D): three-leveled y-x-z hierarchy of tiles (see Section 3.2) with each tile composed by a series of 2D TIFF files;
- TIFF (tiled, 3D): three-leveled y-x-z hierarchy of tiles (see Section 3.2) with each tile composed by a series of multipage (3D) TIFF files;
- TIFF (tiled, 4D): four-leveled c-y-x-z hierarchy of tiles (see Section 3.2) with each tile composed by a series of multipage (3D) TIFF files;
- Vaa3D raw: single Vaa3D raw file containing a 3D image;
- Vaa3D raw (series, 2D): a folder containing a series (1+) of 2D Vaa3D raw files;
- Vaa3D raw (tiled, 3D): three-leveled y-x-z hierarchy of tiles (see Section 3.2) with each tile composed by a series of Vaa3D 3D raw files;
- Vaa3D raw (tiled, 4D): four-leveled c-y-x-z hierarchy of tiles (see Section 3.2) with each tile composed by a series of Vaa3D 3D raw files.

For time series, the user should check the Time series of option, provided that the time points are stored according to the level t specification of the TeraFly format (see Section 3.2).

Once the input path is inserted, simply pressing ENTER on the keyboard will import the inputted volume and prepare for the conversion step, which brings up the following updated dialog.



In the output form, the user should select the output format from the drop-down menu, and provide the output path where the image files will be stored.

Supported output formats are:

- TIFF (tiled, 2D): ℓ-t-y-x-z hierarchy of tiles (see Section 3.2) with each tile composed by a series of 2D TIFF files;
- TIFF (tiled, 3D): ℓ-t-y-x-z hierarchy of tiles (see Section 3.2) with each tile composed by a series of multipage (3D) TIFF files;
- TIFF (tiled, 4D): ℓ-t-c-y-x-z hierarchy of tiles (see Section 3.2) with each tile composed by a series of multipage (3D) TIFF files;
- Vaa3D raw (tiled, 3D): ℓ-t-y-x-z hierarchy of tiles (see Section 3.2) with each tile composed by a series of Vaa3D 3D raw files;
- Vaa3D raw (tiled, 4D): ℓ-t-c-y-x-z hierarchy of tiles (see Section 3.2) with each tile composed by a series of Vaa3D 3D raw files.

For all the output formats considered, the t level is not inserted in the output hierarchy in case the input image consists of a single time point.

Other advanced options, which we suggest to leave at their default values, are:

- Resolutions: these are the resolution scales to be produced at level ℓ of the hierarchy (see Section 3.2) and are automatically determined by TeraConverter following the approach described at the beginning of this section. Optionally, the user can select which resolutions have to be produced, or add more to the ones already in the list;
- Tile dims: these are the individual 3D image tile dimensions T_x, T_y, and T_z (see Section 3.2). Optionally, the user can set them to values higher than the actual image dimensions (e.g. $2000 \times 2000 \times 2000$ in the example considered) to produce a nontiled pyramid, i.e. a pyramid with just one big y-x-z image block;
- downsampling method: the method used to generate the downsampled resolution scales of the pyramid. Supported methods are: mean(2x2x2) that computes the average intensity for each $2 \times 2 \times 2$ image block , and max(2x2x2) that computes the maximum intensity in the same block;

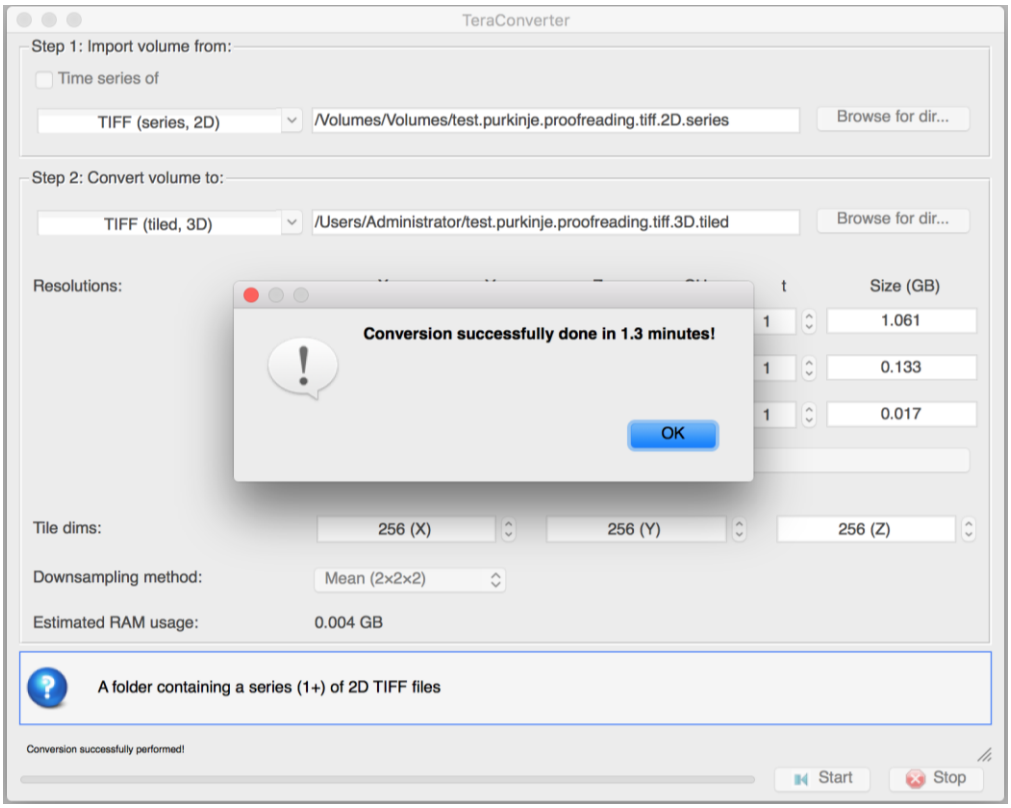
At the bottom of the conversion form, TeraConverter shows in the Estimated RAM usage field a precise estimate of the computer memory that will be used during the conversion process. This is computed as follows:

$$Estimated\ RAM\ usage\ (gigabytes) = \frac{\dim_x \times \dim_y \times \dim_c \times bpp \times 2^{k^*}}{8 \times 10^9}$$

where $\dim_x$ and $\dim_y$ are the input image size along x and y, $\dim_c$ is the number of channels, bpp is the number of bits per pixel (usually 8 or 16), and k^* is the highest resolution scale index among those that have to be produced ($k^* = 0$ if only the original resolution scale is selected).

Once ready, the user can start the conversion by pressing the Start button. The progress bar displayed at the bottom of the file dialog will show the status of the process along with an estimate

of the remaining time. When the conversion ends, a message will inform the user and provide the overall time elapsed (see screenshot below).



3.4 Benchmarks

We have benchmarked export to TeraFly's format `TIFF(tiled,3D)` using default export settings (see Section 3.3) from datasets of different sizes (see Table 3.1) in order to provide users with rough estimates of export times for their own datasets. Moreover, to demonstrate the efficiency of our conversion tool, we compared processing time, computer memory usage, and output image size to that of BigDataViewer when converting the same datasets using similar settings.

For BigDataViewer, we used the following procedure as suggested by Pietzsch et al. (2015):

- from ImageJ, we opened a dataset using the `Virtual Stack` option. Specifically, for series of 2D TIFF files, we used `File > Import > Image Sequence...` and checked the `Virtual Stack` option, whereas for single multipage 3D TIFF files, we used `File > Import > TIFF Virtual Stack`;
- from ImageJ, we launched the BigDataViewer conversion tool using `Plugins > BigDataViewer > Export Current Image as XML/HDF5`;
- from the `Export for BigDataViewer` file dialog, we always left the default options (automatic `mipmap setup` and `use deflate compression` on) before launching the conversion process.

All benchmarks were repeated at least twice for each experiment, and they were run on a MacBook Pro Retina (early 2015) with 2.5 GHz Intel Core i7 (4 cores), 16 gigabyte RAM, 500 gigabyte Solid State Drive (SSD). All benchmarks used Java version 1.8.0_66 with 8 GB RAM maximum heap size

as in Pietzsch et al. (2015). To minimize the input/output conflicts, the input images were read from an external 2.0 terabyte USB 3.0 hard drive, whereas the output image were saved on the internal MacBook SSD drive. The benchmark results are summarized in Table 3.2.

Dataset	**Dimensions** ($x \times y \times z \times c$)	**Image format**
Purkinje cells 1 [(a)]	1,020 × 1,020 × 1,020 × 1	multipage 3D compressed TIFF (8 bits, grayscale)
Purkinje cells 2 [(a)]	1,020 × 1,020 × 1,020 × 1	series of 2D compressed TIFFs (8 bits, grayscale)
Purkinje cells 3 [(a)]	1,951 × 2,122 × 608 × 1	multipage 3D compressed TIFF (8 bits, grayscale)
Purkinje cells 4 [(a)]	1,951 × 2,122 × 608 × 1	series of 2D compressed TIFFs (8 bits, grayscale)
Rat neuron gray [(b)]	9,640 × 6,952 × 179 × 1	series of 2D compressed TIFFs (8 bits, grayscale)
Rat neuron color [(b)]	9,640 × 6,952 × 179 × 3	series of 2D compressed TIFFs (24 bits, RGB)
Mouse cerebellum [(a)]	8,249 × 3,662 × 3,646 × 1	series of 2D compressed TIFFs (8 bits, grayscale)

Table 3.1. 3D/4D image volumes used to test the dataset export performance. (a) Acquired using Confocal Light Sheet Microscopy *(Silvestri, et al., 2012)* for the Human Brain Project (courtesy of F.S. Pavone, L. Sacconi, L. Silvestri). (b) Acquired using 2-photon microscopy (courtesy of R. Tisen).

Experiment	**Raw data size**	**Output pyramid image size** [(a)]		**Memory usage**		**Export time**	
		TC	BDV	TC	BDV	TC	BDV
Purkinje cells 1	1.06 GB	0.62 GB (L=3)	0.59 (L=3)	0.04 GB	1.87 GB	1 m	2 m
Purkinje cells 2	1.06 GB	0.62 GB (L=3)	0.59 (L=3)	0.04 GB	8.21 GB	1 m	18 m
Purkinje cells 3	2.52 GB	1.23 GB (L=5)	1.20 GB (L=5)	0.07 GB	3.89 GB	2 m	4 m
Purkinje cells 4	2.52 GB	1.23 GB (L=5)	1.20 GB (L=5)	0.07 GB	8.17 GB	2 m	29 m
Rat neuron gray	12.00 GB	5.44 GB (L=5)	6.09 (L=7)	1.07 GB	8.63 GB	11 m	27 m
Rat neuron color	35.99 GB	15.63 GB (L=5)	error [(b)]	3.22 GB	error [(b)]	30 m	error [(b)]
mouse cerebellum	110.14 GB	28.97 GB (L=6)	30.14 GB (L=7)	0.97 GB	8.61 GB	89 m	201 m

Table 3.2. Benchmark results for TeraConverter (TC) and BigDataViewer (BDV) when exporting datasets of different sizes and formats. (a) L denotes the number of pyramid levels. (b) the message was "*only 8, 16, and 32-bit images are currently supported*"

On average, TeraConverter was 7x faster and 74x more memory efficient than BigDataViewer, and yielded a slightly better compression ratio (0.44 compared to 0.45 of BigDataViewer). In two cases (Purkinje cells 2 and Purkinje cells 4) BigDataViewer was particularly slower, perhaps due to an issue triggered by the different input format (multipage 3D TIFF). In another case (Rat neuron color) BigDataViewer displayed an error message indicating that it currently does not support export from color images (see Table 3.2). However, even after excluding these challenging yet important use-cases, TeraConverter, on average, was still 2x faster and 30x more memory efficient than BigDataViewer, thus demonstrating the efficiency of our conversion tool.

Supplementary Note 4. Instant zoom-in

4.1 Introduction

Throughout the exploration of the volumetric image, TeraFly translates user mouse drags and scrolls into Volumes of Interest (VOIs) to be displayed from the appropriate pyramid layer (**Supplementary Note 2**). Whereas a multiresolution pyramid data structure is necessary for effective data accessing, it alone would not allow to achieve real-time performance when a true 3D rendering (e.g. real-time maximum intensity projection or alpha blending) is considered. Thus, in this supplement we propose two critical techniques that make TeraFly much faster. These two techniques are a robust random-access VOI estimation method and an effective fetch-and-display strategy that gives the clue that the system has instantly responded to the request to find a 3D VOI, while the full image content is loaded quickly.

4.2 Mean-shift of mean-shift (MSMS)

We propose a "mean-shift of mean-shift" (MSMS) method to accurately estimate 3D VOIs that a human user likes to see when the user is zooming-in or out the displayed contents in a 3D viewer of Vaa3D. In the context of TeraFly, MSMS replaces the built-in "Virtual Finger" algorithms (referred as 'bVF' below) of Vaa3D (Peng, et al., 2014), which map users' inputs in the 2D plane of a computer screen to the 3D locations of bio-entities (for example, cells, neurons or microtubules) in the volumetric space of a 3D image stack. We noted that bVF algorithms might not be robust to produce accurate VOIs when there were a lot of background noises in the displayed image area.

The MSMS algorithm is described as follows. First, we hypothesize that the user is zooming-in on a region containing one or more bio-entities which appear brighter than the surrounding background. Thus, we define a circle of radius d centered on the viewport's center, and then randomly generate n 2D points (*seeds*) within it. Some seeds will fall on (or very close to) the displayed bio-entities, whereas others will fall on the background. Then, for each seed, we consider the intensity profile along the shooting ray orthogonal to the screen, and apply the mean-shift (MS) method (Fukunaga & Hostetler, 1975) to find the mode of the intensity distribution. The MS algorithm begins by finding the center of mass (CoM) of the projection ray (see Algorithm 1, line 4), and then repeatedly reestimates a CoM using progressively smaller intervals around the proceeding CoM until convergence (see Algorithm 1, lines 5-11). Here, we hypothesize that the point to which MS converges will provide the missing third coordinate to map the 2D seed, which is defined on the screen, to the 3D point in the volumetric image. When there are multiple color channels, MS is applied to each color channel separately and the 3D location is estimated by finding the one with the maximal intensity among candidates detected for all colour channels independently. After this step, all the n 2D seeds have been mapped to the corresponding 3D locations. However, only the 3D points corresponding to the bio-entities displayed on the foreground will form a dense cluster, whereas the others will fall on random background locations. To robustly estimate the center of this cluster from all the 3D points, we apply a formulation of the MS algorithm suited for clustering. At every algorithm iteration, each 3D point shifts to the CoM calculated in a sphere of radius r (see Algorithm 1, lines 23-30). The algorithm ends when there no more shifts are detected (see Algorithm 1, line 31), i.e. when all 3D points converge to the center

of the densest cluster of 3D points. Intuitively, this can be viewed as finding the mode of a data point distribution, with data points belonging to the three-dimensional Euclidean space. Finally, we calculate the VOI as a box centered on the mode with size $B_x \times B_y \times B_z$, which corresponds to the 3D viewer size (default value is $256(x) \times 256(y) \times 256(z)$). We typically used $n = 20$, $d =$ 10% of the viewport diagonal, and $r = 100$.

We evaluated the accuracy of the MSMS based VOIs, compared to those generated based on both human observation and bVF. In the scale of 1 (wrong) to 10 (perfect), MSMS had an average score 9.6 over 100 trials, which was nearly perfect compared to what a user would expect. On the contrary, bVF achieved an average score 7 in similar comparison. MSMS outperformed bVF especially on the lower-resolution layers of the images with low contrast and SNR (Fig. 4.2). Next, we compared the computation time of a 3D VOI with MSMS and bVF for six different image stacks of various sizes (Fig. 4.1b). Remarkably, the computation time for MSMS was well below 10 milliseconds in all the test cases as for bVF, thus indicating we successfully eliminated one substantial bottleneck in terabytes data visualization without introducing any human detectable delay.

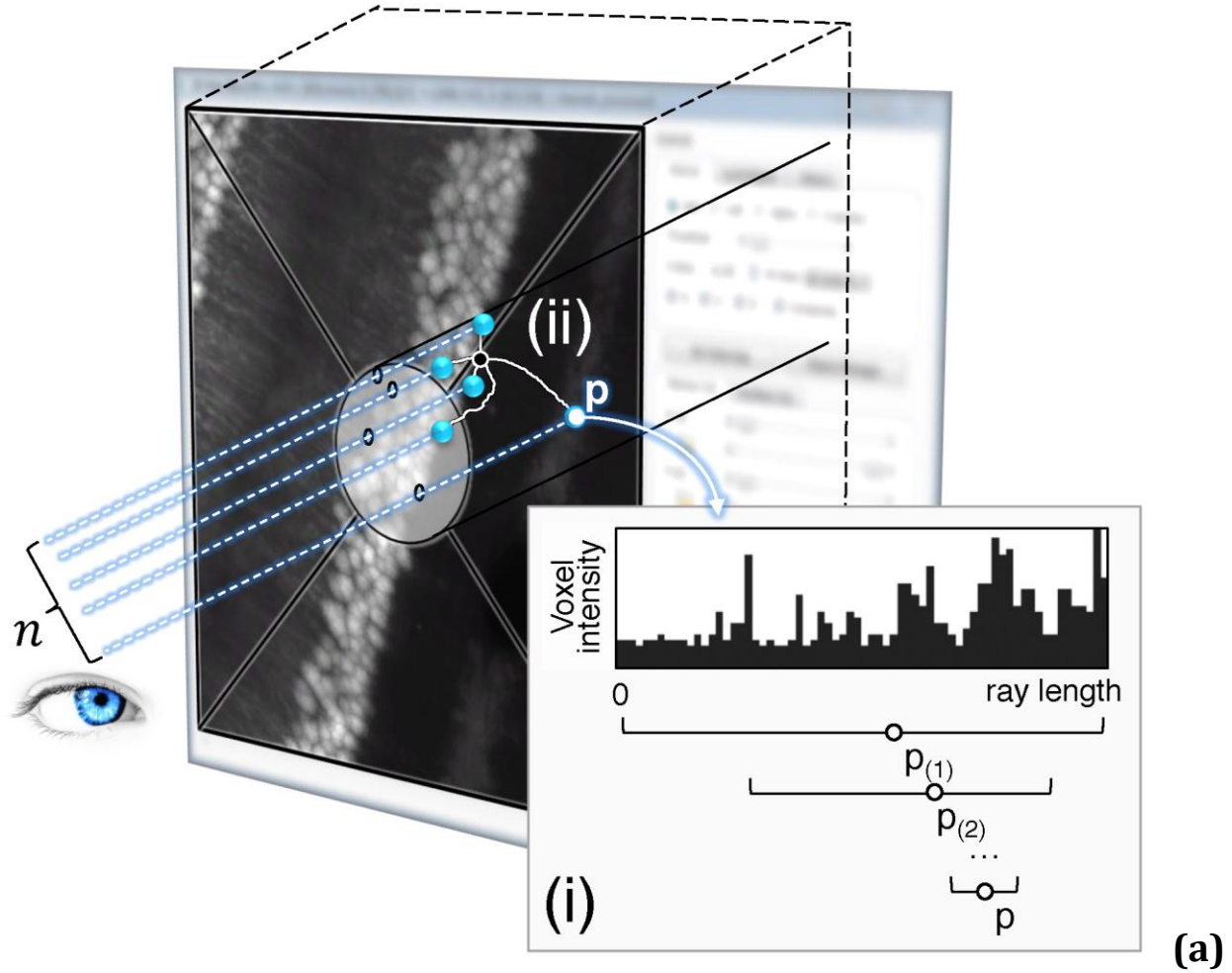
4.3 Fetch-and-display strategy

We developed a "fetch-and-display" method to ensure a very fast response from TeraFly when new data was visualized. The goal for this method was that when the mouse scrolls and zooms-in to a VOI, the higher resolution image content of such VOI will be retrieved from the file system and displayed as quickly as possible for fast and smooth exploration of the image. We accomplished this task by performing, in parallel, two different operations: (i) reusing part of the currently viewed content to display instantly a linearly interpolated preview of the VOI; and (ii) loading from the pyramid image only the blocks that contain the required image content. More formally, let be VOI^{ℓ} be the VOI currently viewed that belongs to the pyramid image level ℓ ($VOI^{\ell} \equiv I^{(k)}$ when the exploration starts, with $I^{(k)}$ being the highest pyramid image level in a pyramid of $k + 1$ layers) and $VOI^{\ell'}$ the VOI to be retrieved and displayed from the higher resolution layer ℓ', with $\ell' \leq \ell$ (usually $\ell' = \ell - 1$). Then, TeraFly interpolates to the resolution of ℓ' the portion of VOI^{ℓ} that intersects with $ROI^{\ell'}$ and displays the result. This gives the clue that the system has correctly and instantly responded to the request, and also makes more acceptable to wait for the full image to be loaded. Meanwhile, in parallel, TeraFly loads the image content from the blocks of the pyramid image layer ℓ' that intersect with $VOI^{\ell'}$ and updates the display with the higher resolution data.

Algorithm 1 MSMS

$s_i \in \mathbb{R}^2$:	a 2D *seed* point randomly taken in a circle of radius d centered on the viewport's center
$S \in \mathbb{R}^{2\times n}$	$S = \begin{pmatrix} s_1 & s_2 & \cdots & s_n \end{pmatrix}$
$Q \in \mathbb{R}^{3\times n}$:	$Q = \begin{pmatrix} q_1 & q_2 & \cdots & q_n \end{pmatrix}$, with $q_i = \begin{pmatrix} s_i \\ p_i \end{pmatrix}$, $p_i \in \mathbb{R}$
$p_i^{(k)}$:	p_i at kth mean-shift iteration
$\mathcal{R}$:	n intensity distributions along the shooting rays associated to the points s_i
$\mathcal{R}_i$:	ith distribution of $\mathcal{R}$
$\mathcal{R}_{i[a,b]}$:	ith distribution of $\mathcal{R}$ considered in the interval $[a, b]$
$\mathrm{CoM}(\cdot)$:	center of mass of a given distribution
Qs:	3D points in Q shifted after one mean-shift iteration
qs_i:	ith element of Qs
r:	mean-shift flat kernel radius

procedure MSMS($S, \mathcal{R}$)
  $i \leftarrow 1$
  while $i \leq n$ **do** ▷ First Mean-Shift
    $p_i^{(1)} \leftarrow \mathrm{CoM}(\mathcal{R}_i)$ ▷ initialize p_i with the center of mass of $\mathcal{R}_i$
    $L^{(1)} \leftarrow |\mathcal{R}_i|$ ▷ the initial search range spans the entire ray length
    $k \leftarrow 1$
    repeat
      $k \leftarrow k + 1$
      $L^{(k)} = 0.7 \times L^{(k-1)}$ ▷ narrow down the search range
      $p_i^{(k)} \leftarrow \mathrm{CoM}\left(\mathcal{R}_{i\left[p_i^{(k-1)} - \frac{1}{2}L^{(k)}, p_i^{(k-1)} + \frac{1}{2}L^{(k)}\right]}\right)$
    until $p_i^{(k)} \neq p_i^{(k-1)}$
    $i \leftarrow i + 1$
  end while
  $Qs \leftarrow Q$
  $shifted \leftarrow true$
  while $shifted$ is $true$ **do** ▷ Second Mean-Shift
    $shifted \leftarrow false$
    $i \leftarrow 1$
    while $i \leq n$ **do**
      $N \leftarrow 0$ ▷ counter
      $qs_i \leftarrow \vec{0}$
      $j \leftarrow 1$
      while $j \leq n$ **do**
        if $||q_j - q_i|| \leq r$ **then** ▷ consider a sphere of radius r
          $qs_i \leftarrow qs_i + q_j$
          $N \leftarrow N + 1$
        end if
        $j \leftarrow j + 1$
      end while
      $qs_i \leftarrow qs_i / N$ ▷ compute CoM in a sphere of radius r
      if $||qs_i - q_i|| > \vec{0}$ **then**
        $shift \leftarrow true$ ▷ detect a shift
      end if
      $i \leftarrow i + 1$
    end while
    $Q \leftarrow Qs$
  end while
return q_1 ▷ all q_i are equal as they converged to the distribution mode
end procedure



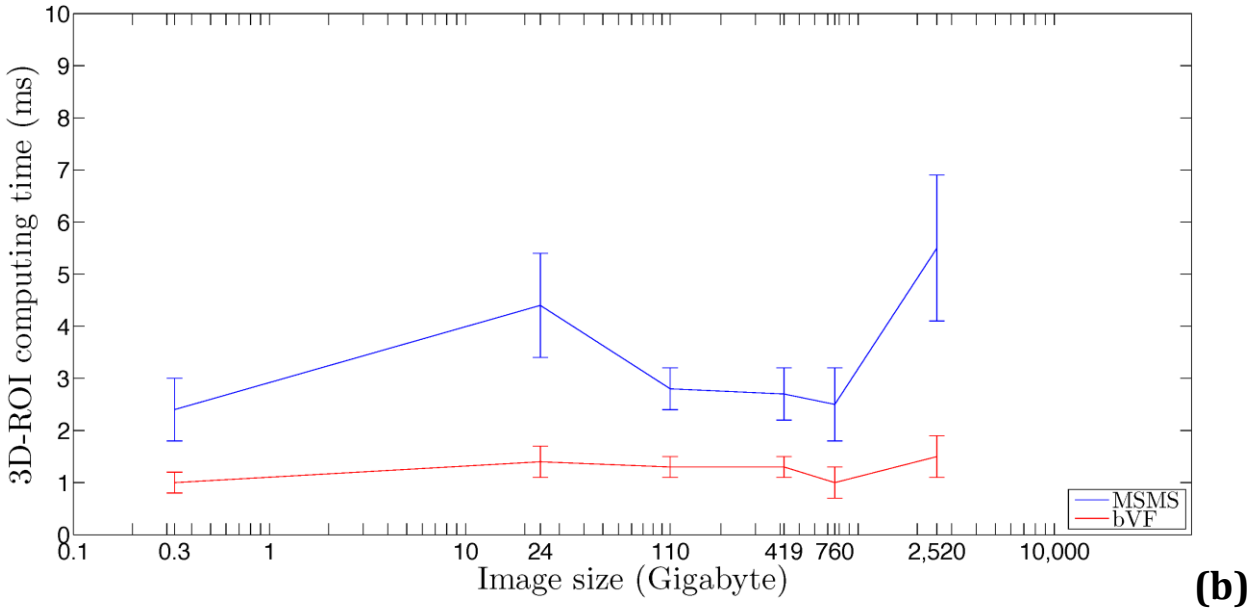


Fig. 4.1. (a) Schematic illustration of the 3D VOI estimation based MSMS. (i) First, the mean-shift method is applied on the voxel intensity along each of the n shooting rays for the calculation of the corresponding bio-entity 3D locations. (ii) Then, the mean-shift strategy is again applied on these 3D locations for the calculation of the VOI center. (b) Average 3D-ROI computing time with MSMS and bVF for six 3D/4D image stacks with size ranging from 0.3 gigabytes to 2,520 terabytes (see Table 2.1).

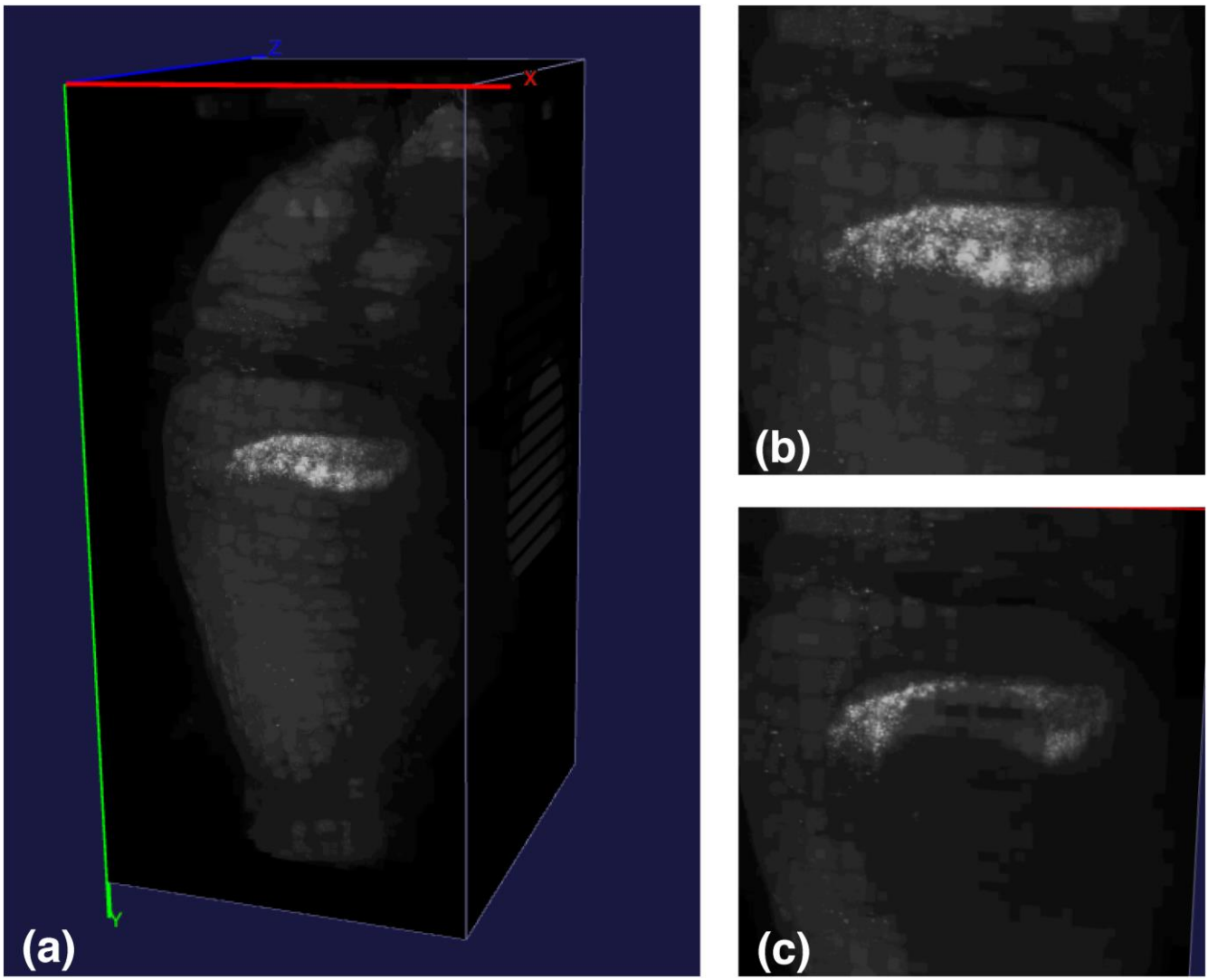


Fig. 4.2. (a) The lowest-resolution layer of a whole mouse brain image in which only a few cells of the hippocampus express Green Fluorescent Protein (GFP). (b) Zoom-in using MSMS. (c) Zoom-in using bVF (Peng, et al., 2014).

Supplementary Note 5. User Guide

5.1 Introduction

TeraFly is a tool developed on top of Vaa3D (Peng, et al., 2010) to allow real-time (i.e. subsecond) visualization and assisted analysis of terabytes of multidimensional volumetric images. The tool has been implemented in C++ with Qt and OpenGL and it is freely and publicly available both as open-source and as binary package along with the main Vaa3D distribution.

TeraFly comes with an open data format based on a multiresolution tiled pyramid image that enables fast access of small parts (tiles) of an image volume at different resolution scales. This data format is based on the open-standard TIFF format (Adobe Developers Association, 1992), and consists of a set of image files organized into a hierarchy of $n + 1$ levels of folders (n being the data dimensionality, e.g. for 3D datasets $n = 3$ up to $n = 5$ for 5D datasets). The complete specification of this data format is described in the **Supplementary Note 3** along with the procedure to export a dataset to the TeraFly's format using our tool TeraConverter (included in Vaa3D).

This supplement describes common use cases for TeraFly and the TeraFly software itself (installation, user interface, available functionalities, etc.). Since TeraFly is in ongoing development, this supplement might not include the most recent updates, or it could be based on a slightly different naming of the various functionalities available. In this supplement, we use the version 2.1.0 of the TeraFly software.

5.2 Installation

From versions 2.0.0 and later, TeraFly is directly integrated into Vaa3D. Thus, it is only required to install Vaa3D (versions 3.*) to be able to use TeraFly. The procedure for installing Vaa3D and launching TeraFly is described as follows:

1. go to http://vaa3d.org.
2. click on the 'Download' tab.
3. from the pull-down menu, choose the appropriate program corresponding to your operating system (Mac, Linux or Windows) and download it to your local computer. The Vaa3D program is often compressed as a ZIP file and you need to unzip it before use.
    - for Linux and Windows, unzip the program into a new folder before use.
    - for Mac, unzip the program to generate an installer program for standardized installation. Double-click the installer program to launch a GUI of installation, and enter the user password to complete the installation, which will put the Vaa3D program under the /Applications/vaa3d folder on the local computer.
4. run the Vaa3D program to launch the GUI.
    - for Windows, double-click the program vaa3d_msvc.exe.

Nature Methods doi:10.1038/nbt.3767

- for Linux, on a command-line console, enter the folder that contains the unzipped program and type ./start_vaa3d.sh command to run.
- for Mac, double-click the program /Applications/vaa3d/vaa3d64.app (without displaying the log information) or on the command line console run the command /Applications/vaa3d/vaa3d64.app/Contents/MacOS/vaa3d64 to start the GUI with the running log information displayed.

5. launch TeraFly from

Vaa3D Menu > Advanced > Big-Image-Data > TeraFly

5.3 User Interface

TeraFly has a friendly and usable User Interface (UI) designed to maximize the image visualization area while providing a number of components and functionalities to boost 3D visualization and annotation. The overall scheme of the UI is depicted in Fig. 5.1. In the following subsections, we describe in detail each component of the TeraFly's UI. The components marked as '*advanced*' correspond to advanced use cases, whose description is out of the scope of this supplement.

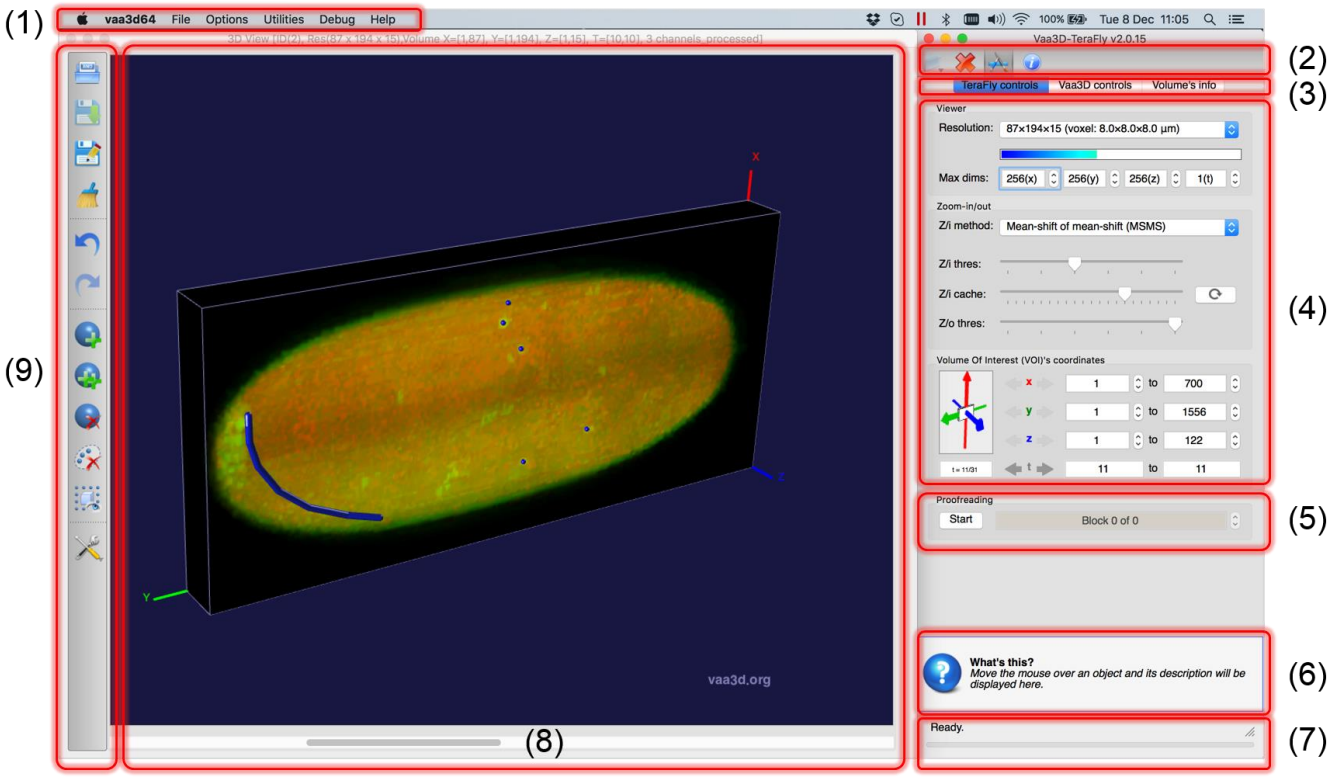


Fig. 5.1. The graphical user interface of Vaa3D-TeraFly that has a number of components (1)~(9). (1) Menu bar. (2) Menu toolbar. (3) Tab switch. (4) TeraFly's exploration controls. (5) TeraFly's proofreading controls. (6) Interactive help box. (7) Status bar. (8) Vaa3D 3D viewport and visualization controls. (9) 3D object annotation toolbar.

5.3.1 Menu bar

The TeraFly's menu bar contains the following menus:

TeraFly Menu > File

- open a TeraFly volume
- open a HDF5 (BigDataViewer) volume
- open a recent volume
- close the current volume
- load 3D annotations from a .ano file
- save 3D annotations to a .ano file

TeraFly Menu > Options

- Import : volume import options (*advanced*)
  - regenerate metadata files mdata.bin and cmap.bin (see Section 3.2)
  - regenerate volume map file vmap.bin (see Section 3.2)
- Annotations : annotation display options
  - Markers : marker display options
    - size
    - VOI extra margin size (*advanced*)
  - Curves : curve display options
    - aspect (3D tube / 2D skeleton)
    - skeleton width
  - virtual space size (auto / unlimited) (*advanced*)
- Navigation : image exploration options (*advanced*)
  - Shifts : directional shifts options (*advanced*)
  - Fetch-and-display : fetch-and-display strategy (preview/streaming/direct) (*advanced*)

TeraFly Menu > Utilities

- Annotations > Markers : annotation utilities for 3D markers (e.g. cells)
  - Convert : export to Vaa3D/TeraFly .apo from other formats (e.g. VTK, MaMuT)
  - Analyze : analyze / compare .apo annotation files
- Time-series : generation of time series (*advanced*)

TeraFly Menu > Debug

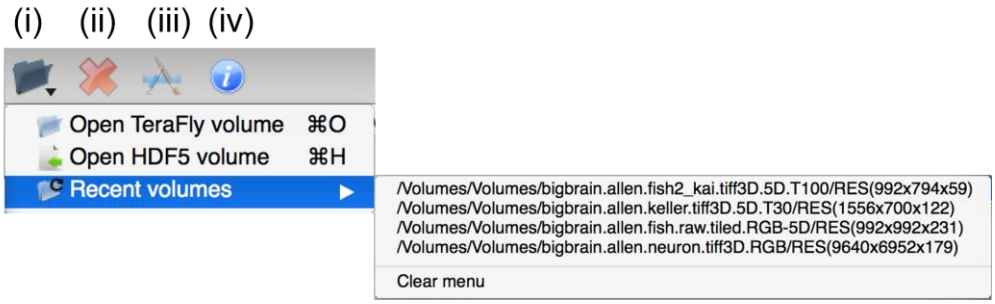
- show message log (*only for developers*)
- set message verbosity level (*only for developers*)

TeraFly Menu > Help : shows version info and changelog.

On MacOS, the menu bar is automatically integrated into the OS menu bar. On other operative systems (e.g. Windows, Linux), the menu bar is right above the Menu toolbar (Section 5.3.2).

5.3.2 Menu toolbar

The menu toolbar is described as follows:



(i) open a new volume (TeraFly or HDF5) through a File Dialog or opens a recently volume from the displayed pull-down menu containing a list of the recently opened volumes.
(ii) close the currently opened volume.
(iii) display or hide the TeraFly's 3D object annotation toolbar (Section 5.3.9).
(iv) display the tool's info and changelog.

5.3.3 Tab switch

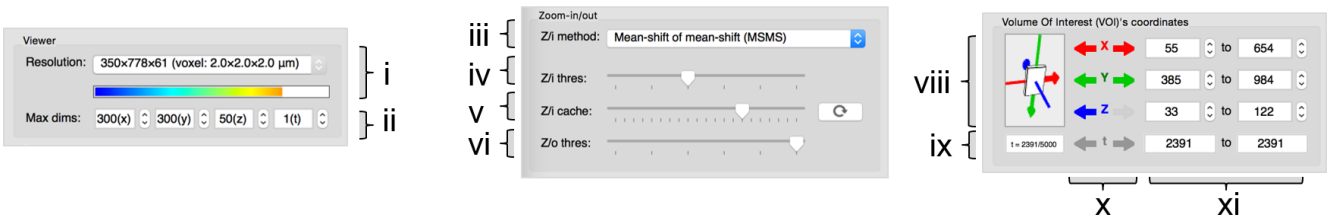
The tab switch is described as follows:

- TeraFly controls: shows the TeraFly controls (Section 5.3.4 and 5.3.5);
- Vaa3D controls: shows the Vaa3D controls (Section 5.3.8);
- Volume's info: shows a panel containing information on the currently opened image (e.g. size, voxel dimensions, tile dimensions).

5.3.4 TeraFly exploration controls

The TeraFly's exploration controls are grouped into three categories: (a) "*Viewer*", to set up the 3D viewer dimensions and image resolutions; "*Zoom-in/out*", to set up advanced options for the mouse-scroll Google-Earth like zoom-in and zoom-out; and "*Volume Of Interest (VOI)'s coordinates*", to check and specify the portion of the volume to be displayed.

Individual controls are described as follows.



(i) *(advanced)* image resolution composed by a pull-down menu and a heat map like bar. From the pull-down menu, the user can select the resolution at which he/she wants to display the

currently viewed volume / currently selected volume of interest (VOI). The heat map like bar indicates the currently displayed image resolution (the "hotter", the higher);

(ii) set the Vaa3D 3D viewer *x*-*y*-*z* (space) and *t* (number of time points) dimensions, i.e. the amount of image data displayed at the time. The larger, the more graphic card memory is used and the slower is the visualization. Suggested range is [100,300] for *x*-*y*-*z* and [1-10] for *t*;

(iii) *(advanced)* choose from pull-down menu the method used to generate the volume of interest (VOI) when zooming-in with the mouse scroll;

(iv) *(advanced)* set the zoom-in threshold value (see *Vaa3D controls* > *Zoom & Shift* > *Zoom*, Section 5.3.8) to trigger the image resolution increase when zooming-in with the mouse scroll. The default is 50. Set it to 100 to disable this feature, so that TeraFly will never increase the image resolution when zooming-in;

(v) *(advanced)* set the zoom-in cache sensitivity value. This corresponds to the overlap between the VOI to be displayed and the VOI already in the cache required to instantly recover and display the VOI from the cache, instead of loading new data from the storage. Sensitivity of 0% means that TeraFly will always use the cache, whereas 100% means that TeraFly will always load the image data from the storage;

(vi) *(advanced)* set the zoom-out threshold value (see *Vaa3D controls* > *Zoom & Shift* > *Zoom*, Section 5.3.8). Similarly to the zoom-in threshold value, this changes the zoom factor (see *Vaa3D controls* > *Zoom & Shift* > *Zoom*, Section 5.3.8) that triggers the image resolution decrease when zooming-out with the mouse scroll. The default value is 0. Set it to -100 to disable this feature, so that TeraFly will never decrease the image resolution when zooming-out;

(vii) reset all zoom-in/out controls to default values;

(viii) interactive 3D reference system. During 3D visualization, this indicates how the displayed image is oriented in the 3D space and allows the user to change the volume orientation by dragging the displayed cube with the mouse towards the desired direction. In proofreading mode, this indicates the position of the displayed block in the whole image (Section 5.3.5);

(ix) indicates the currently displayed time point and the total number of available time points *(5D data only)*;

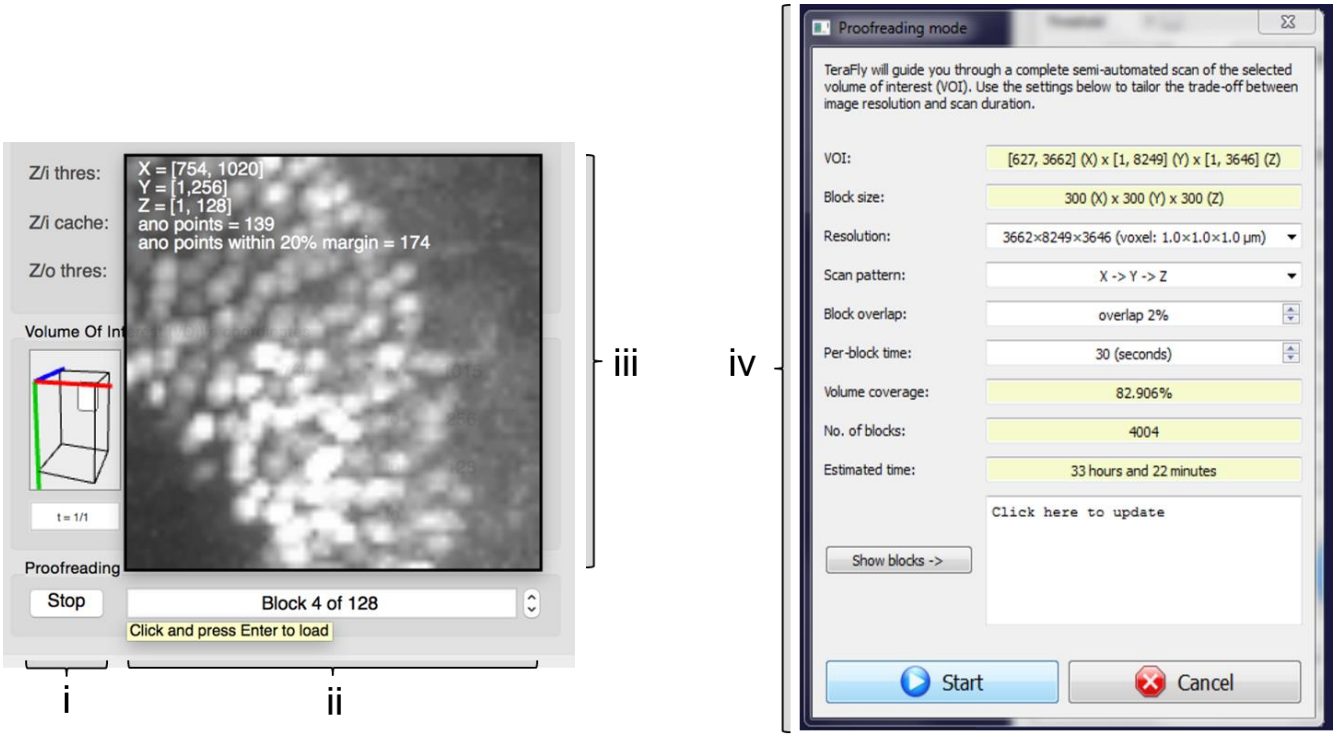
(x) directional shifts. These buttons (one pair for every axis *x*-*y*-*z*-*t*) allow the translation of the 3D viewer throughout the image along the corresponding axes by an amount that can be changed from *Options* > *Navigation* > *Directional shifts* (see Section 5.3.4(x)). Default is 50% for *x*-*y*-*z* and 0% for *t*;

(xi) *(advanced)* specify the volume of interest absolute spatial coordinates, i.e. referred to the highest resolution image. This corresponds to use the Vaa3D volume-cut scrollbars and can be used to specify a VOI to be displayed at a higher resolution (i.e. using the *Resolution* controls, see (i)) or to be analyzed in proofreading mode.

5.3.5 TeraFly proofreading controls

TeraFly's proofreading modality allows to perform a stoppable/resumable block-by-block scan of the entire volume (or a selected VOI) to proofread automatic cell counts or neuron reconstructions. When the proofreading modality is on, it is not possible to change the image resolution with zoom-in/zoom-out or by using the directional shifts (Section 5.3.4(x)).

The proofreading panel UI's elements are depicted and described as follows.



(i) start or terminate the proofreading session.

(ii) *QuickScan*. Scrolling the mouse wheel in this area allows instant inspection of the blocks to roughly check hundreds of blocks per minute and load only the nonempty ones. Pressing "Enter" loads the currently viewed block.

(iii) maximum intensity projection along *z* of the block being scanned. It also displays the block coordinates and the number of annotations points (corresponding to cells or neuron segments), to allow the user deciding whether to load (or skip) this block.

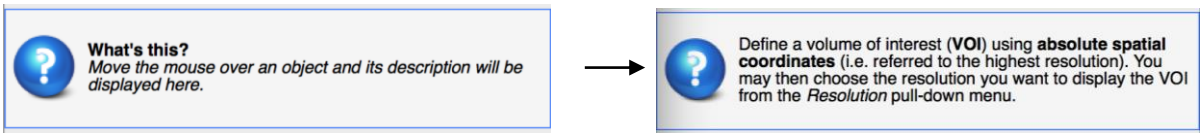
(iv) dialog displayed when a proofreading session starts (see (i)). It allows to choose the following settings to set up the proofreading session:

- *VOI*: the volume of interest to proofread. This can be modified using the Vaa3D volume cut scrollbars controls (Section 5.3.8) or the TeraFly's VOI inputs (Section 5.3.4(xi));
- *Block size*: block dimensions along x, y, and z. They can be modified using the TeraFly's 3D viewer dimensions (Section 5.3.4(ii)). The larger the block, the fewer blocks will be needed for the block-by-block scan of the selected VOI;

- *Resolution*: the resolution scale at which the image should be scanned. The coarsest the resolution, the fewer the blocks needed for the scan of the VOI;
- *Scan pattern*: the pattern that defines the scan-path. For instance, "*X -> Y -> Z*" means to move along the *x* axis first and, when reached the rightmost block, move along the *y* axis and, when reached the bottommost block, move along the *z* axis, and so on;
- *Block overlap*: the overlap (in percentage) between two adjacent blocks. Some overlap is usually needed to avoid missing errors in the boundary regions;
- *Per-block time*: the inputted estimate of per-block analysis time, which serves to calculate the overall estimated time for the entire proofreading session (see last point of this list);
- *Volume coverage*: the coverage (in percentage) of the selected VOI with respect to the whole image;
- *No. of blocks*: the calculated total number of blocks. This depends on the *VOI*, *Block size*, *Block overlap*, and *Resolution* inputs;
- *Estimated time*: the overall estimated time for the entire proofreading session. This is calculated as *Per-block time* × *No. of blocks*.

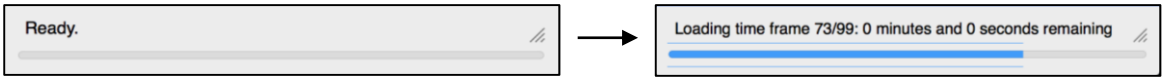
5.3.6 Interactive helpbox

This text box displays helpful information on the various components of the TeraFly's UI. Just moving the mouse over one of these components will trigger the corresponding description to appear in the box (hence the name "interactive helpbox").



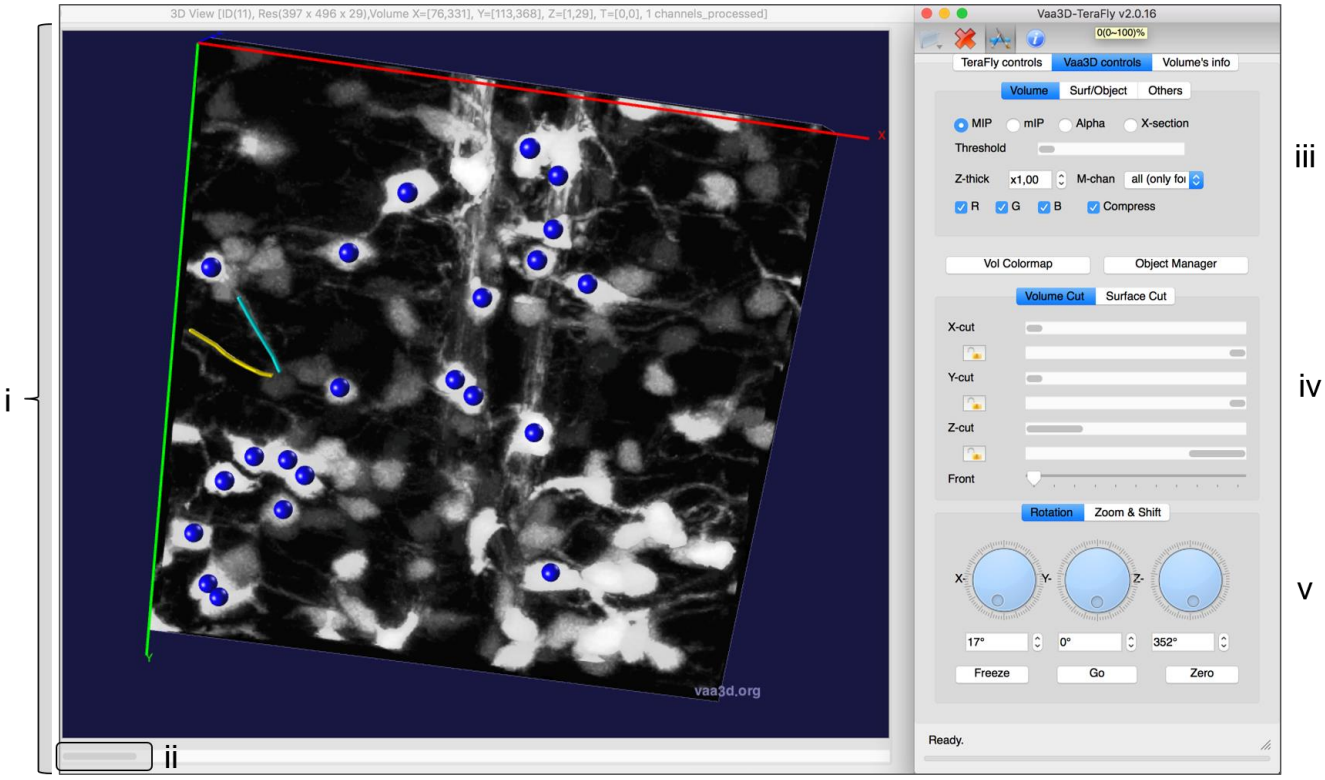
5.3.7 Status bar

The TeraFly's status bar displays the information (e.g. kind of operation, progress percentage, remaining time) related to the operation currently performed in background. Usually, TeraFly is so fast that the update of the status bar cannot even be noticed. Only when the operation involves a massive amount of input/output operations (such as when visualizing hundreds of time points at the time, see Section 5.3.4(ii)) the status bar updates can be noticed.



5.3.8 Vaa3D 3D viewport and visualization controls

We provide here the most important user guidelines for the Vaa3D viewport/3D renderer and for the Vaa3D visualization controls that can be accessed from the TeraFly's *Tab Switch* (Section 5.3.3). For more details, please visit the official Vaa3D website at http://vaa3d.org.



(i) 3D interactive viewer: visualize and explore the image data in 3D smoothly using:

- rotation: hold the mouse left button and move towards the desired rotation direction.
- zoom: mouse scroll down (zoom-in), mouse scroll up (zoom-out) or double-click to the desired location (only in TeraFly).
- shift: use the arrow keys (more precise) or press SHIFT + mouse left button to drag the image towards the desired direction.

To access the useful *Virtual Finger*-powered 3D object annotation tools (Peng, et al., 2014), right-click on the image and choose the desired functionality from the pop-op menu. Some examples are:

- "*1-right-stroke to define a marker*" to define a marker with one mouse right-stroke directly on the displayed object (e.g. a cell).
- "*1-right-stroke to define a 3D curve*" to define a 3D curve with one mouse right-stroke along a displayed linear structure (e.g. a neurite).
- "*Zoom-in HighRezImage: 1-right-stroke ROI*" to zoom-in to the ROI defined with one mouse right-stroke.

(ii) time scrollbar: scroll through the *t* axis and select the desired time point (5D data only).

(iii) rendering controls: change the image/surface objects rendering options. This includes "*Vol Colormap*", a useful dialog to adjust the image LUT.

(iv) volume cut scrollbars: specify the volume of interest to display.

(v) rotation, zoom, and shift controls: these are alternative ways to control rotation, zoom and shift without using mouse drags and scrolls.

5.3.9 3D object annotation toolbar

The TeraFly's 3D annotation toolbar consists of a set of tools designed to boost the visualization-assisted analysis and proofreading of the displayed image and surface data (e.g. cells, neuron reconstructions). These tools are described in the following table.

	Open a .ano linker file. A .ano file contains a list (one or more) of annotation files supported by Vaa3D (i.e. .apo, .marker, .swc).
	Save 3D surface objects (i.e. cell markers, point clouds, tubules) to the already opened .ano file (it will be overwritten).
	Save annotations to a new .ano file.
	Deletes all 3D surface objects in the image (cannot be undone).
	Undo the last 3D object edit action.
	Redo the last 3D object edit action.
	Add one (or more) 3D marker(s) with one right-click.
	Add one (or more) 3D marker(s) with two right-clicks (the two clicks should be made from two different viewing angles).
	Remove one (or more) 3D marker(s) with one right-click. Only one marker at the time can be removed.
	Remove multiple 3D markers with one right-stroke. All the 3D markers within the contour drawn with the mouse will be removed.
	Show / hide the 3D markers around the displayed volume of interest (VOI). The outer margin size can be set from *Options* > *Annotations* > *Markers* -> *Virtual margin size* (see Section 5.3.1) Default is 20% of the VOI.
	A shortcut to access to the *Options* menu (see Section 5.3.1).

5.4 Use cases

In this section we provide a step-by-step guideline for common use cases which require little or no previous training or experience with our TeraFly software. When needed, we will refer to our test data repository (link) for the download of a specific dataset.

5.4.1 Opening a TeraFly dataset

(i) download and unzip one of the test datasets available from the our test data repository.

(ii) launch Vaa3D and then launch TeraFly from the menu '*Advanced*' > '*Big-Image-Data*' > '*TeraFly*'.

(iii) from the TeraFly menu bar (Section 5.3.1) or from the menu toolbar (Section 5.3.2) click on the '*Open TeraFly volume*' button. From the file dialog, select *any* of the volume folders starting with "RES" (e.g. RES(255x255x255)). Each of these folders stores a different volume resolution (**Supplementary Note 3**, Section 3.2). Whatever folder is chosen, TeraFly will display the one corresponding to the lowest resolution image.

5.4.2 Image exploration

(i) open a dataset (Section 5.4.1).

(ii) in the 3D viewer window (Section 5.3.8(i)), press and hold the left mouse button to rotate the 3D-rendered image freely.

(iii) in the 3D viewer window, hold the 'Shift' key, and then press and hold the left mouse button to shift around the 3D-rendered image freely.

(iv) in the 3D viewer window, use the mouse scroll wheel to zoom in or zoom out on the 3D-rendered image freely. You may notice when TeraFly switches to the higher (or to the lower) resolution image just as Google Earth does. The higher the resolution, the smallest portion of the volume is displayed.

(v) in the 3D viewer window, from the menu accessible by right-clicking on the image, select '*Zoom-in HighRezImage: 1-right-stroke ROI*' to zoom-in to the VOI defined with one right-stroke.

(vi) in the 3D viewer window, use the time scrollbar to move along the time points (5D data only)(Section 5.3.8 (ii)).

(vii) in the TeraFly window, select the "*Vaa3D controls*" tab from the Tab switch (Section 5.3.3) and then click on the "*Vol Colormap*" to change brightness/contrast (Section 5.3.8 (iii)). From the same dialog, you can change other visualization options, such as rendering (*MIP*, *mIP*, *alpha*, *X-Section*) and *z*-thickness.

(viii) in the TeraFly window, select the "*TeraFly controls*" from the Tab switch (Section 5.3.3) and then use the directional shifts (Section 5.3.4(x)) to allow the translation of the 3D viewer along the corresponding axes.

5.4.3 3D annotation of biological structures

We describe in this section how to generate 3D markers and 3D curves for the displayed biological structures (e.g. cells, neurites, vessels) and how to curate (modify, annotate, delete, save/load) such data directly in the volumetric space of a 3D image stack (hence the term '3D annotation'). Applications of these annotation tools include (but are not limited to) cell counting, neurite tracing (e.g. in a whole brain image), vessel tracing, the generation a "gold standard" to feed semi- or fully-supervised image analysis algorithms and the proofreading of the output of these algorithms.

5.4.3.1 Generation and curation of 3D markers

(i) download and unzip the mouse.cerebellum.1GB.zip dataset (courtesy of F.S. Pavone, L. Sacconi, L. Silvestri) from our test data repository and open it in TeraFly (Section 5.4.1).

(ii) adjust brightness/contrast as needed (Section 5.4.2).

(iii) zoom-in to the higher resolution scales until the cells can be clearly seen.

(iv) from the TeraFly 3D object annotation toolbar displayed on the left (Section 5.3.9), activate the "*1-right-click to define a marker*"( ) tool to input 3D markers directly on the image with one mouse right-click. For a more precise cells pinpointing, activate the "*2-right-click to define a marker*"( ) tool to input 3D markers directly on the image with two mouse right-clicks (from two different viewing angles).

(v) zooming-in or out to other resolution scales will also scale the size of the displayed markers, while maintaining their absolute position in the image space. To change the size of the displayed markers, go to '*Options*' > '*Annotations*' > '*Markers*' > '*Size*' (Section 5.3.1).

(vi) to remove one marker at the time, activate the "*1-right-click to delete a marker*"( ) tool and right-click on the marker to be deleted.

(vii) to remove multiple markers at the time, activate the "*1-right-stroke to delete a group of markers*"( ) tool and draw a curve with one mouse right-stroke around the markers to be deleted. Hold the 'SHIFT' key to delete only the markers along the curve.

(viii) to undo/redo, click on the and buttons, respectively.

(ix) to input text annotations for a specific marker, right-click on it to activate the pop-up menu for that marker and select the first menu entry "*Marker #N...*". This will bring up a dialog where the annotation text can be inserted.

(x) to change the color of a specific marker, right-click on it to activate the pop-up menu for that marker and select the "*Color*" entry.

(xi) to save the markers and related annotations to a new .ano/.apo file, use the "*Save annotation as*"( ) tool and select the output filename from the file dialog.

(xii) to save the markers and related annotations to the already imported .ano/.apo file, use the "*Save annotations*"( ) tool (it will overwrite the .apo file).

(xiii) to load the markers and related annotation from an existing .ano/.apo file, use the "*Load annotations*"( ) tool and select the input file from the file dialog. For instance, you might want to open the somas-subvol.ano (already present in the test dataset) which contains the automated cell counts for the entire volume.

(xiv) to delete all the markers in the image, use the "*Clear annotations*" ( ) tool.

5.4.3.2 Generation and curation of 3D curves

(i) download and unzip the rat.neuron.zip dataset (courtesy of R. Tisen) from our test data repository and open it in TeraFly (Section 5.4.1).

(ii) adjust brightness/contrast as needed (Section 5.4.2(vii)).

(iii) from the coarsest resolution scale image (i.e. the one displayed when the image volume is opened), right-click on the image to activate the pop-up menu and select '*1-right-stroke to define a 3D curve (ver 2a)'* to trace linear structures (e.g. neurites) directly on the image with one right-stroke. At this coarse resolution, only the thicker neurites can be traced.

(iv) creating a 3D curve will also enable the *curve editing mode*, which activates additional options from the right-click pop-up menu. To exit from the *curve editing mode*, select the "*finish editing this neuron*" option from the right-click pop-up menu. To re-enter the *curve editing mode*, right-click on the curve tree to be edited and select the "*edit this neuron*" option.

(v) to input text annotations for a specific curve segment, right-click on it to activate the pop-up menu for that segment and select the first menu entry "*Neuron/line #N…*". This will bring up a dialog where the annotation text can be inserted.

(vi) to change the color of a specific curve segment, right-click on it to activate the pop-up menu for that segment and select the "*Color*" entry.

(vii) to change the aspect (tube/skeleton) of the displayed curves, go to '*Options*' > '*Annotations*' > '*Curves*' > '*Aspect*' (Section 5.3.1).

(viii) to change the thickness (radius) of a specific curve segment (*curve editing mode* on), right-click on it to activate the corresponding pop-up menu and select the "*change nearest neuro-segment radius*" option.

(ix) to smooth a specific curve segment (*curve editing mode* on), right-click on it to activate the corresponding pop-up menu and select the "*deform the neuron-segment*" option. This will open a file dialog from which the value "*resolution step*" should be increased.

(x) to delete one or more neuron segments (*curve editing mode* on), right-click on the image and select the "*delete multiple neuron-segments by stroke*" option from the pop-up menu. Then hold the 'SHIFT' key and draw a contour with one mouse right-stroke around the curve segments to be deleted. Release the 'SHIFT' key to delete only the curve segments intersecting the contour drawn.

(xi) to undo/redo, click on the and buttons, respectively.

(xii) use the TeraFly zoom-in and zoom-out functions combined with the annotation tools previously described to trace neurites of different thickness at different resolution scales. For instance, thin neurites can only be clearly seen (and thus traced) at the highest resolution scale, whereas thick neurites can only be traced at the coarsest resolution scale (they are too thick at the highest resolution scale), and so on.

(xiii) to save the curve segments and related annotations to a new .ano/.swc file, use the "*Save annotation as*"( ) tool and select the output filename from the file dialog.

(xiv) to save the curve segments and related annotations to the already imported .ano/.swc file, use the "*Save annotations*"( ) tool (it will overwrite the .swc file).

(xv) to load the curve segments and related annotation from an existing .ano/.swc file, use the "*Load annotations*"( ) tool and select the input file from the file dialog. For instance, you might want to open the neuron-traced.ano (already present in the test dataset) which contains the manual tracing for the entire volume.

(xvi) to delete all the curve segments in the image, use the "*Clear annotations*" ( ) tool.

5.4.3.3 Fast proofreading of automatic analysis outputs in large image volumes

(i) download and unzip the mouse.cerebellum.1GB.zip dataset (courtesy of F.S. Pavone, L. Sacconi, L. Silvestri) from our test data repository and open it in TeraFly (Section 5.4.1).

(ii) adjust brightness/contrast as needed;

(iii) set the 3D viewer *x*-*y*-*z* dimensions (Section 5.3.4(ii)) to 300(*x*) x 300(*y*) x 300 (*z*). This will correspond to the dimensions of a single image block during the block-by-block proofreading of the entire volume.

(iv) from the TeraFly 3D object annotation toolbar displayed on the left (Section 5.3.9), use the "*Load annotations*"( ) tool and select the somas-subvol.ano file that comes along with the dataset. This will import the automatic cell counts and display the detected cells as 3D markers.

(v) from TeraFly, click on the '*Start*' button in the '*Proofreading*' panel (Section 5.3.5). A dialog will be shown (Section 5.3.5(iv)) from which to set proofreading session parameters (we suggest block overlap of 20%, and the highest resolution scale). Then, press the '*Start*' button to enter the proofreading mode.

(vi) once the first block is loaded and displayed, use the *QuickScan* (Section 5.3.5(ii)) feature to jump to the next nonempty block by scrolling the mouse wheel up and down on the block's spinbox in the '*Proofreading*' panel and then click and press *Enter* to load the block.

(vii) to quickly proofread the cell locations in the current block, use the tools in the TeraFly's 3D annotation toolbar as described in Section 5.4.3.1.

(viii) to visualize the markers outside the displayed VOI and thus to avoid errors (i.e. false positives and false negatives) in the VOI's boundary regions, use the "*Show/hide markers around the displayed VOI*" tool (). The markers outside the VOI will be displayed in white color and cannot be modified nor deleted from this VOI (indeed they rely to the adjacent VOIs). Alternatively, one can choose a higher block overlap (e.g. 40%) at step (v), but at the cost of an increased number of blocks to proofread.

(ix) use the "*Save annotations*"() button in the TeraFly's toolbar to save the corrected cells.

(x) to exit the proofreading mode, click on the '*Stop*' button in the '*Proofreading*' panel.

5.4.4 Automated analysis of the annotation results

We describe in this section the tools available in TeraFly to quantitatively analyze the manually inputted or automatically generated 3D object annotations. Applications of these tools include (but are not limited to) the quantitative comparison of "gold standard" annotations to computer (or human) generated annotations or, further, the quantitative comparison of annotations curated/proofread with different tools (**Supplementary Note 6**). Of note, comparing multiple ($n > 2$) instances of image annotations, such as cell counts generated by multiple experts or algorithms, would be computationally cumbersome without a look-up table designed for efficient representation and search of such annotation data. In TeraFly, we use an octree data structure to encode the 3D object coordinates that allows real-time search of the annotation data (**Supplementary Note 7**).

5.4.4.1 Counting and/or labeling quasi-coincident 3D markers

Given a set A of $|A|$ markers m_i, $i = 1, \dots, |A|$, with each marker m_i associated to a 3D point (x_i, y_i, z_i), and a tolerance distance d_{max}, we define two markers m_i and m_j as *quasi-coincident* ($m_i \cong_{d_{max}} m_j$) if (and only if) their distance is less or equal to d_{max}, i.e.:

$$m_i \cong_{d_{max}} m_j \quad \text{iff } d(m_i, m_j) \leq d_{max}$$

To simplify the notation, we will omit the subscript d_{max} from the notation '$\cong_{d_{max}}$', and thus indicate two quasi-coincident markers m_i and m_j with $m_i \cong m_j$.

In TeraFly, there are two options to calculate quasi-coincident 3D markers in a .apo file:

(i) from the TeraFly menu (Section 5.3.1), go to '*Utilities*' > '*Annotations*' > '*Markers*' > '*Count duplicates in the whole image*' and insert the tolerance distance d_{max}. TeraFly will count the quasi-coincident markers in the whole image currently opened.

(ii) from the TeraFly menu, go to '*Utilities*' > '*Annotations*' > '*Markers*' > '*Label duplicates in .apo file*' and choose the input .apo file, the output .apo file, and the tolerance distance d_{max}. TeraFly will label the quasi coincident markers with white color and save them (along with the other markers) to the selected output file.

5.4.4.2 Compare/diff multiple annotation files

Given the sets of markers $M_1, \dots, M_n$ corresponding to n distinct annotation files (.apo) generated by n distinct annotators (either humans or computer algorithms), we want to obtain the subset D that contains only the markers where at least two annotators disagree. Thus, we want to calculate:

$$D = \bigcup_i D_i \quad where \quad D_i = \{m \in M_i : \exists j \neq j : x \notin M_j\}$$

The procedure to do this in TeraFly is described as follows:

(i) from the TeraFly menu (Section 5.3.1), go to '*Utilities*' > '*Annotations*' > '*Markers*' > '*Diff of multiple .apo files*' and select any number of input .apo files and the output .apo file.

(ii) TeraFly will calculate $D = \bigcup_i D_i$ and associate a unique color and name to the markers of D_i. This way, it will still be possible to distinguish the various D_i in D. The unique name is extracted from the input .apo filename.

5.4.4.3 Calculate type I and type II errors from two annotation files

Given the sets of markers G and F corresponding to two distinct annotation files (.apo), where G is assumed as gold standard, we want to compare G and F and calculate the type I errors (false positives (FP)) and type II errors (false negatives (FN)) in F. Following the definition of *quasi coincident* markers of Section 5.4.4.1, we want to calculate:

$$FP = \{x \in F : x \ncong y, \ \forall y \in G\}$$

$$FN = \{y \in G : y \ncong x, \ \forall x \in F\}$$

The procedure to do this in TeraFly is described as follows:

(i) from the TeraFly menu (Section 5.3.1), go to '*Utilities*' > '*Annotations*' > '*Markers*' > '*Count type I/II errors from two .apo files*' and select the input .apo files corresponding to G and F and the output .apo file.

(ii) TeraFly will ask to input the tolerance distance d_{max} (Section 5.4.4.1) and, optionally, the unique name identifier of the markers in F that have to be compared with the markers in G. If no name filter is provided, all the markers in F will be compared to all the markers in G.

(iii) TeraFly will store in the output .apo file the two sets FP and FN with the false positive markers labeled with red color and annotated as 'false_positive' and the false negative markers labeled with blue color and annotated as 'false_negative'.

Supplementary Note 6. Annotation of Big-Image-Data

6.1 Introduction

TeraFly enabled us to annotate massive amount of biological images as we demonstrate in this supplement in two real case scenarios (Section 6.2 and Section 6.3). Further, in Section 6.4 we provide benchmark results and a quantitative comparison with other tools demonstrating that TeraFly is both more efficient and more precise in the annotation of biological structures.

6.2 Quantification of Purkinje cells in a 110 GB mouse cerebellum

The first real test case was the precise quantification of Purkinje cells in a L7-GFP whole mouse cerebellum. We ran the Brain cell finder tool (Frasconi, et al., 2014) to localize the Purkinje cells in the 110.1 gigabyte image of the cerebellum of an L7-GFP mouse (Table 2.1). A total of 224,221 locations were detected and saved into a standard ASCII .vtk file to be imported into TeraFly (Section 5.3.1). Then, an expert accurately proofread the cell locations in one quarter (26 gigabytes) of the image (**Supplementary Video 3**) using the TeraFly's proofreading modality (Section 5.3.5 and Section 5.4.3.3). This activity was divided into 16 sessions in which the expert analyzed 1,274 image stacks of size $300(x) \times 300(y) \times 300(z)$ with overlap of 20%. Remarkably, the total time employed was only 10.1 hours, corresponding to about 24 minutes per gigabyte. Assuming the obtained cell count as a gold standard, we then evaluated the performance of the computerized analysis as follows. A predicted cell center $\vec{c}$ was considered to be a true positive (TP) if it matched a gold standard center $\vec{g}$ such that $||\vec{c} - \vec{g}|| = 0$. Unmatched predictions were counted as false positives (FP) and unmatched gold standard centers were counted as false negatives (FN). To this end, we used the type I/II error counting functionality available in TeraFly (Section 5.4.4.3). We finally computed precision (P), recall (R), and F_1 measure as $P = TP/(TP + FP)$, $R = TP/(TP + FN)$, and $F_1 = \frac{2PR}{P+R}$. The performance achieved was $F_1 = 0.98$, which applied to the entire image yields an estimate of 220, 800 Purkinje cells in the whole mouse cerebellum. This result is consistent with previous estimates based on stereology (Woodruff-Pak, 2006; Biamonte, et al., 2009) and is the most complete and precise map of its kind ever obtained.

6.3 Tracing of mammalian neurons

Currently TeraFly has been deployed in several applications to reconstruct (trace) very complicated, large, 3D mammalian neuron morphology from images acquired using different imaging modalities. While a comprehensive application study is beyond the scope of this technical paper, here we describe an example to use TeraFly to annotate rat neuron image acquired using 2-photon microscopy (courtesy of R. Tisen). The image had size $6{,}952(x)\times 9{,}640(y)\times 179(z)$ voxels and two color channels. The produced pyramid image consisted of four additional downsampled images of size from $3{,}476(x)\times 4{,}820(y)\times 80(z)$ to $434(x)\times 602(y)\times 11(z)$. Starting from the lowest-resolution image, an expert accurately traced the thickest neurites. Thinner neurites were traced at the intermediate resolutions, whereas the thinnest ones, which were only partially or not visible at the lowest resolutions, could only be traced at the highest resolutions (**Supplementary Video**

6). With TeraFly visualization and multi-scale one mouse-stroke 3D curve creation (Section 5.4.3.2), the entire neuron could be mapped efficiently.

6.4 Benchmarks

From the same whole cerebellum image used in the experiment of Section 6.2, we extracted a 1 GB nonempty nonproofread image of size $1020(x) \times 1020(y) \times 1020(z)$ voxels and compared TeraFly's 3D-based proofreading (Section 5.3.5 and Section 5.4.3.3) with analogous annotation functionalities offered by other image visualization softwares. Specifically, we compared TeraFly to CellCounter/ImageJ (De Vos, 2010; Abrmoff, et al., 2004) and to MaMuT/BigDataViewer (Tinevez & Pietzsch, 2015; Pietzsch, et al., 2015), which are two popular and freely available tools for image visualization and annotation.

To obtain a rough estimate of the proofreading errors (type I and type II) made with the tools considered, we designed a procedure that requires minimal human effort (at least two proofreaders) and no prior definition of a gold standard annotation (Fig. 6.1). The procedure consists of comparing the proofread cells generated with the different annotation tools and then checking (and correcting) one by one the cells where at least two tools disagree. This potentially yields an underestimation of the proofreading errors when all the tools make the same errors, but still can be a valuable method to compare the proofreading performance of the different tools. In our experiment, we employed three proofreaders: the first two who generated the proofread cells with the tools under comparison, and the third one who carefully checked the cells where at least two tools disagreed and marked the type I and type II errors. To implement this procedure, we used the annotation analysis tools available in TeraFly (Section 5.4.4).

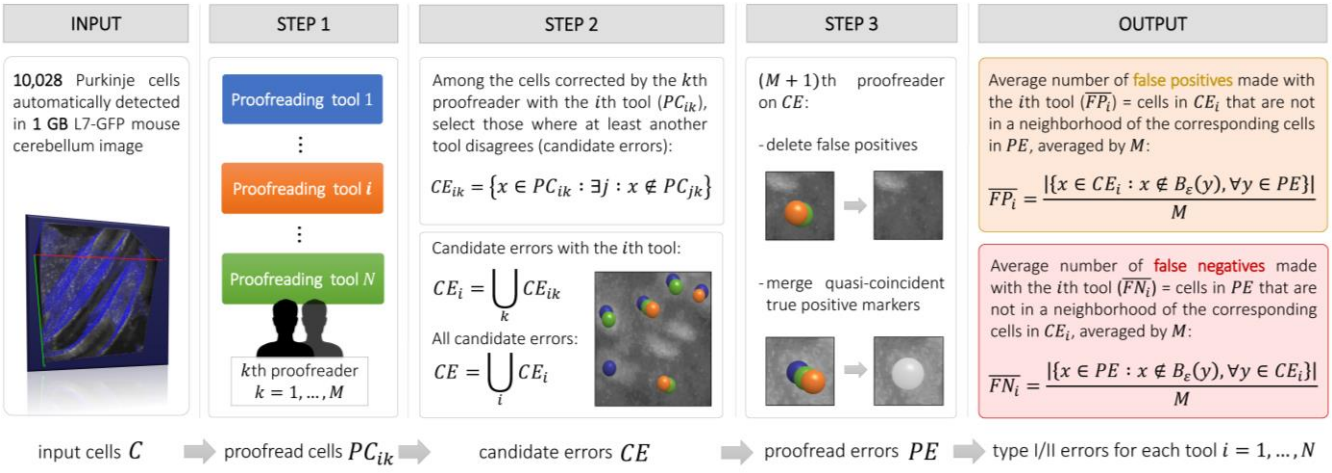


Fig. 6.1. The proposed procedure used to estimate the number of proofreading errors (false positives and false negatives) made with N proofreading tools and that uses $M + 1$ proofreaders ($N = 3$ and $M = 2$ in our experiment).

To generate the proofread cells for the various tools considered, we proceeded as follows. The 1 GB image was virtually divided into 64 blocks of size $300(x) \times 300(y) \times 300(z)$ (overlap 60 voxels), that were proofread separately. Such subdivision was a good trade-off between the number of blocks to proofread (the fewer, the faster the proofreading) and the number of cells in

a single block (the higher, the more difficult and error prone is the proofreading). Then, for each of the tools considered, the first proofreader accurately proofread the 64 blocks after having being trained on additional 5 blocks. In CellCounter/ImageJ, which does not support big-image-data visualization, the 64 image blocks were loaded and visualized one at the time after having been previously extracted from the 1 GB image and saved to 64 image stacks. In TeraFly and MaMuT/BigDataViewer, the entire 1 GB image could be loaded and visualized quickly, thanks to their efficient big-image-data visualization. In TeraFly, we used the proofreading modality with the same settings of the experiment of Section 6.2. In MaMuT/BigDataViewer, we iteratively moved the viewer along x, y and z on the block to proofread.

The results of the proofreading time and type I/II error estimates are reported in Fig. 6.2. On average, proofreading with TeraFly was 4x faster than with Cell Counter/ImageJ and 3x faster than MaMuT/BigDataViewer (Fig. 6.1a). Moreover, TeraFly achieved an average precision $\bar{P} = 0.996$ whereas CellCounter/ImageJ and MaMuT/BigDataViewer achieved $\bar{P} = 0.983$ and $\bar{P} = 0.989$, respectively. These results demonstrate that TeraFly was both considerably faster and more precise than the other tools considered. In particular, TeraFly outperformed the other tools especially on false positives, which were more difficult to detect using a 2D slice-by-slice annotation approach (**Supplementary Video 3-5**).

Finally, we extracted another 64 mostly black image stacks that did not contain any cell. In our case, these formed up to 50% of the whole mouse cerebellum image and thus they could slow down proofreading significantly. This was the case of Cell Counter/ImageJ and MaMuT/BigDataViewer but not of TeraFly (Fig. 6a), as we developed a fast previewing technique that boosted proofreading in case of partially or totally empty stacks (Section 5.3.5).

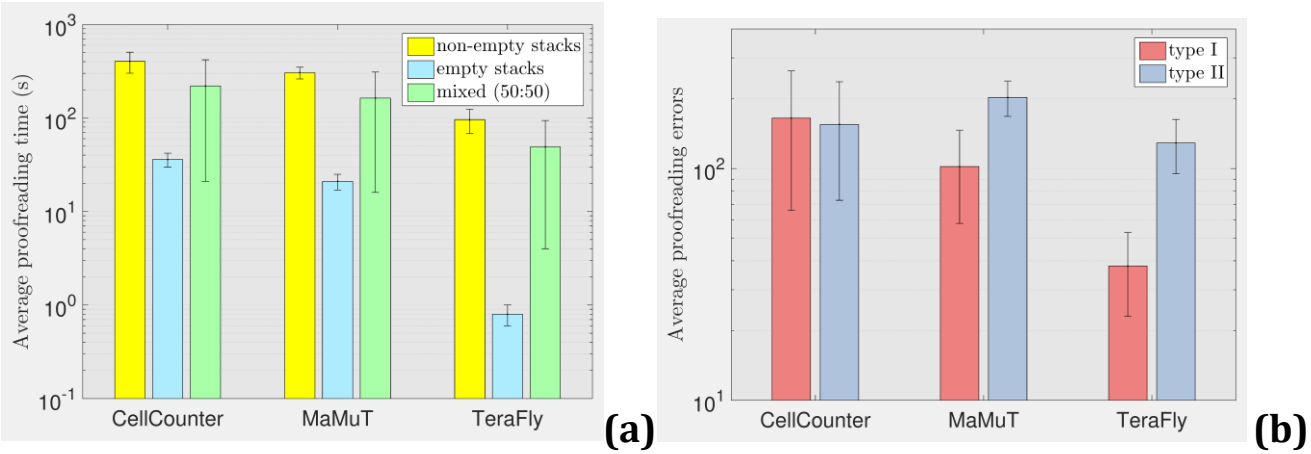


Fig. 6.2. (a) Average time (mean ± s.d.) for proofreading of automated Purkinje cell counts in 64 nonempty 3D image stacks, 64 empty image 3D stacks, and 128 mixed (64 nonempty and 64 empty) 3D image stacks extracted from the L7-GFP 110 gigabyte whole mouse cerebellum image. (b) Proofreading errors (type I/false positives and type II/false negatives) made on the 64 nonempty image stacks of (a).

Supplementary Note 7. 3D object representation

To facilitate very efficient 3D annotation of biological structures (Section 5.4.3), and the automated analysis of such 3D annotations (Section 5.4.4), in TeraFly we used the octree data structure (Meagher, 1982) to encode the 3D objects at runtime. Specifically, we employ a hierarchical 8-ary tree structure in which each node subdivides the space it represents into 8 equally-sized nonoverlapping octants. The root of the octree corresponds to the whole image at the highest resolution (Fig. 7.1(i)), and each $1 \times 1 \times 1$ voxel-sized node stores the point belonging to one or more 3D objects (Fig. 7.1(ii)). Let P be the set of 3D points needed to represent all the 3D objects, this structure has two advantages over a simpler, unordered array of points: (i) the memory required for its representation is still on the order of $|P|$, like for an array; and (ii) the time complexity for finding the objects in a Volume of Interest (VOI) is $O(\log|P|)$ (Meagher, 1982; Narasimhan, et al., 2006), whereas for an array it is $O(|P|)$. Specifically, the VOI query time for 3D objects was always negligible with the octree, and two orders of magnitude smaller than with the array (Fig. 7.2).

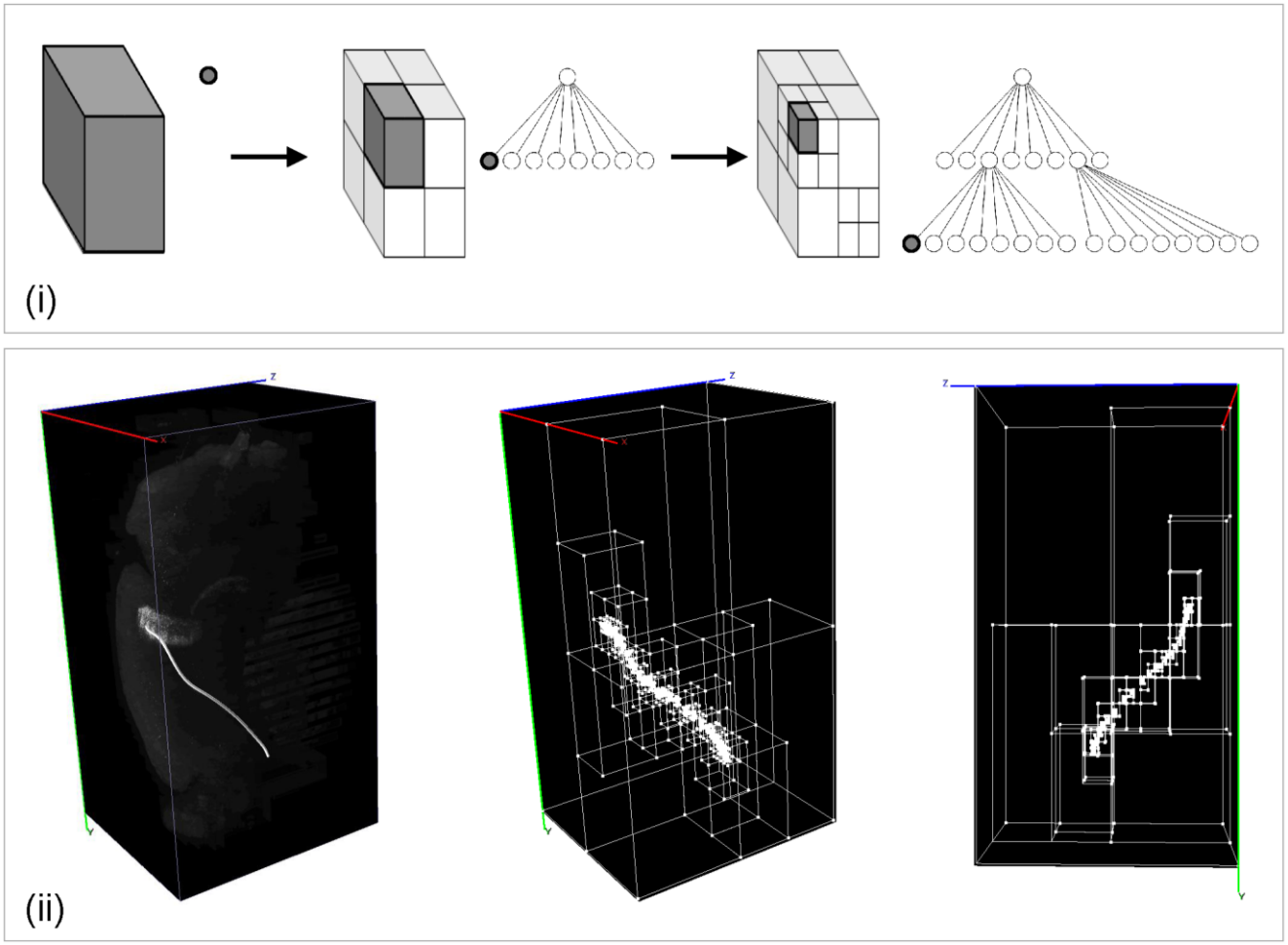


Fig. 7.1. Octree-based representation of the annotated 3D objects. (i) First steps of the generation of an octree. (ii) A neurite traced in a whole mouse brain image and the corresponding octree viewed from two different angles.

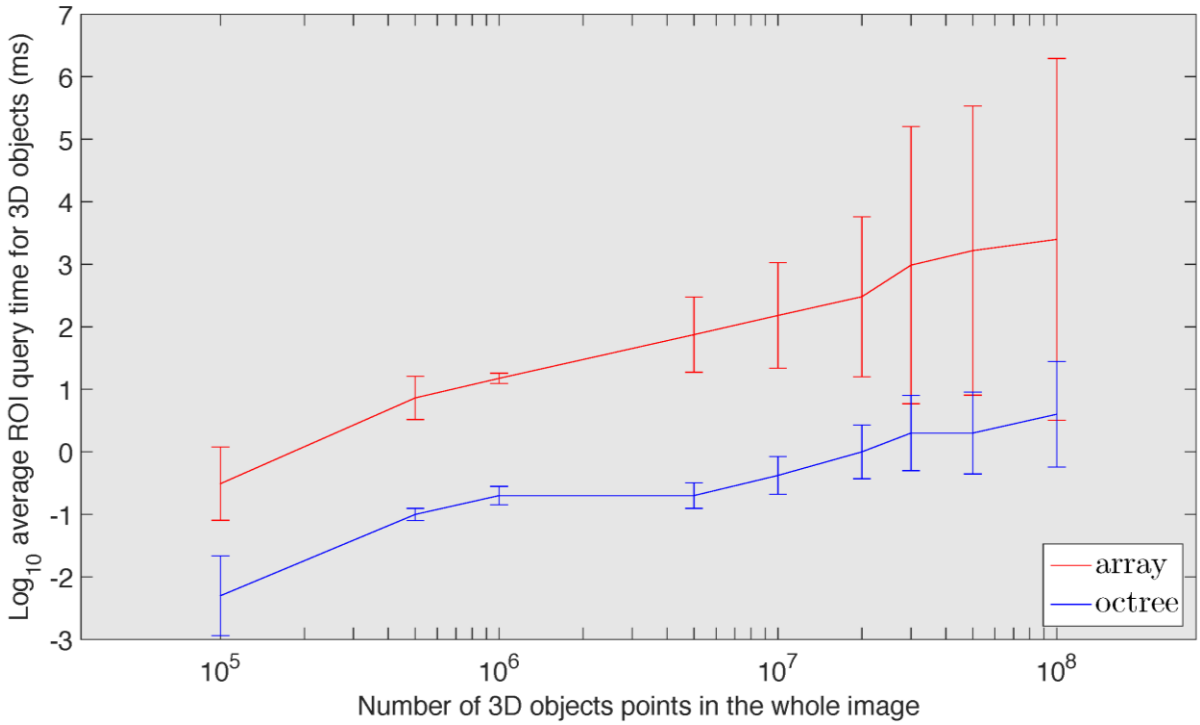


Fig. 7.2. Average query time (mean ± s.d.) for 3D objects in a VOI using octree and array data structures to store 3D objects coordinates. Each data point was obtained from 100 VOIs of size $256(x) \times 256(y) \times 256(z)$ randomly taken from a 1 terabyte image with an increasing number of uniformly distributed 3D markers.

References

Abrmoff, M., Magalhes, P. & Ram, S., 2004. Image Processing with ImageJ. *Biophotonics International,* 11(7), p. 36–42.

Adobe Developers Association, 1992. *TIFF revision 6.0 specification.* [Online]
Available at: https://partners.adobe.com/public/developer/en/tiff/TIFF6.pdf

Amat, F. et al., 2015. Efficient processing and analysis of large-scale light-sheet microscopy data. *Nature Protocols,* 10(11), pp. 1679-1696.

Biamonte, F. et al., 2009. Interactions between neuroactive steroids and reelin haploinsufficiency in Purkinje cell survival. *Neurobiology of Disease,* 36(1), p. 103–115.

Carpenter, A. et al., 2006. Cellprofiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biology,* 7(10), p. R100.

Chung, K. & Deisseroth, K., 2013. CLARITY for mapping the nervous system. *Nature Methods,* 10(6), p. 508–513.

Conrad, C. et al., 2011. Micropilot: automation of fluorescence microscopy-based imaging for systems biology. *Nature Methods,* 8(3), pp. 246-249.

De Vos, K., 2010. *Cell Counter.* [Online]
Available at: http://rsbweb.nih.gov/ij/plugins/cell-counter.html

Frasconi, P. et al., 2014. Large-scale automated identification of mouse brain cells in confocal light sheet microscopy images. *Bioinformatics,* 30(17), p. i587–i593.

Fukunaga, K. & Hostetler, L. D., 1975. The estimation of the gradient of a density function, with applications in pattern recognition.. *Information Theory, IEEE Transactions on,* 21(1), pp. 32-40.

Jeong, W. K. et al., 2010. SSECRETT and neurotrace: Interactive visualization and analysis tools for large-scale neuroscience data sets. *IEEE Comput. Graph. Appl.,* 30(3), pp. 58-70.

Lau, C. et al., 2008. Exploration and visualization of gene expression with neuroanatomy in the adult mouse brain. BMC Bioinformatics, 9(1), 153. *BMC Bioinformatics,* 9(1), p. 153.

Long, F. et al., 2009. A 3D digital atlas of C. elegans and its application to single-cell analyses. *Nature Methods,* 6(9), pp. 667-672.

Long, F., Zhou, J. & Peng, H., 2012. Visualization and analysis of 3D microscopic images. *PLoS Computational Biology,* 8(6), p. e1002519.

Meagher, D., 1982. Geometric modeling using octree encoding. *Computer Graphics and Image Processing,* 19(2), p. 129–147.

Murphy, R. F., 2012. Cellorganizer: Image-derived models of subcellular organization and protein distribution. In: *Computational Methods in Cell Biology.* s.l.:A.R. Asthagiri and A. P. Arkin, pp. 179-193.

Narasimhan, S., Mundani, R.-P. & Bungartz, H.-J., 2006. *An octree and a graph-based approach to support location aware navigation services.* s.l., s.n., p. 24–30.

Peng, H., 2008. Bioimage informatics: a new area of engineering biology. *Bioinformatics,* 24(17), pp. 1827-1836.

Peng, H. et al., 2010. V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. *Nat Biotech,,* 8(4), p. 246–249..

Peng, H. et al., 2014. Virtual Finger boosts three-dimensional imaging and microsurgery as well as terabyte volume image visualization and analysis. *Nature Communications,* Volume 5.

Pietzsch, T. et al., 2012. Fiji: an open-source platform for biological-image analysis. *Nature Methods,* 9(7), p. 676–682.

Pietzsch, T., Preibisch, S., Tomancak, P. & Saalfeld, S., 2012. ImgLib2 - generic image processing in Java. *Bioinformatics,* 28(22), p. 3009–3011.

Pietzsch, T., Saalfeld, S., Preibisch, S. & Tomancak, P., 2015. BigDataViewer: visualization and processing for large image data. *Nature Methods,* 12(6), pp. 481-483.

Pologruto, T., Sabatini, B. & Svoboda, K., 2003. Scanimage: Flexible software for operating laser scanning microscopes. *BioMedical Engineering OnLine,* 2(1), p. 13.

Royer, L. A. et al., 2015. ClearVolume: open-source live 3D visualization for light-sheet microscopy. *Nature Methods,* 12(6), pp. 480-481.

Saalfeld, S., Cardona, A., Hartenstein, V. & Tomancak, P., 2009. CATMAID: collaborative annotation toolkit for massive amounts of image data. *Bioinformatics,* 25(1984–1986).

Silvestri, L., Bria, A., Sacconi, L. & Iannello, G., 2012. Confocal light sheet microscopy: micron-scale neuroanatomy of the entire mouse brain. *Optic Express,* 20(18), p. 20582–20598.

Sommer, C., Straehle, C., Kothe, U. & Hamprecht, F., 2011. *Ilastik: Interactive learning and segmentation toolkit.* s.l., s.n., pp. 230-233.

Stuurman, N. et al., 2010. Computer Control of Microscopes using μManager. In: *Current protocols in molecular biology.* s.l.:John Wiley & Sons, Inc., Hoboken, NJ, USA.

The HDF Group, 2014. *Hierarchical Data Format, version 5.* [Online]
Available at: http://www.hdfgroup.org/HDF5

Tinevez, J.-Y. & Pietzsch, T., 2015. *MaMuT: A Fiji plugin for the annotation of massive, multi-view data.* [Online]
Available at: http://fiji.sc/MaMuT

Tomer, R., Ye, L., Hsueh, B. & Deisseroth, K., 2014. Advanced clarity for rapid and high-resolution imaging of intact tissues. *Nature Protocols,* 7(9), p. 1682–1697.

Walter, T. et al., 2010. Visualization of image data from cells to organisms. *Nature Methods,* 7(3), pp. S26-S41.

Woodruff-Pak, D., 2006. Stereological estimation of Purkinje neuron number in C57BL/6 mice and its relation to associative learning. *Neuroscience,* 141(1), p. 233–243.