BigDataViewer
Visualization and Image Processing for Terabyte Data Sets

Tobias Pietzsch[1], Stephan Saalfeld[2], Stephan Preibisch[3], Pavel Tomancak[1]

[1]Max-Planck Institute of Molecular Cell Biology and Genetics, Dresden, Germany
[2]Janelia Research Campus, Howard Hughes Medical Institute, Ashburn, Virginia, USA
[3]Berlin Institute for Medical Systems Biology, Max Delbrück Center for Molecular Medicine, Berlin, Germany

Abstract—The necessity to make large volumetric datasets available for interactive visualization and analysis has been widely recognized. However, existing solutions build upon proprietary file formats requiring that data are copy-converted before visualization, or use dedicated servers to generate virtual slices that are transferred to client applications, practically leading to insufficient frame rates for truly interactive experience. We present BigDataViewer (BDV), an easily accessible and extensible open source solution for interactive visualization of very large volumes and time series of volumes from both local and remote data sources. Individual image stacks are arbitrarily arranged in global 3D coordinate space, and can be displayed independently or as color composite. The software renders arbitrarily oriented virtual slices through global space, allowing smooth navigation in multi-terabyte image datasets. BDV can be easily extended to handle new data sources such as 3rd party file formats or online data stores. It is re-usable as both visualization frontend and data backend for novel annotation and image processing tools.

I. Introduction

Advances in microscopy today allow live 3D imaging of entire developing embryos with high spatial and temporal resolution, promising new insights in developmental biology. Lightsheet microscopes generate terabytes of data in a matter of a few hours, and it is essential to be able to access and handle these data efficiently. To address this issue, we developed BigDataViewer (BDV), a re-slicing browser for very large multiview image sequences [1]. It is available as a Fiji [2] plugin and integrates seamlessly with Fiji's Multiview Reconstruction pipeline[1].

BDV displays individual image volumes of a multiview, multi-channel, time-lapse data set as transformed (registered) slices in a common global 3D coordinate space. The viewer renders an arbitrarily oriented virtual slice through that global space. It either displays views independently or as color composites. Brightness and color of each view can be adjusted separately. An intuitive user interface allows free translation, rotation, and zoom, as well as moving between timepoints.

1 http://fiji.sc/Multiview-Reconstruction

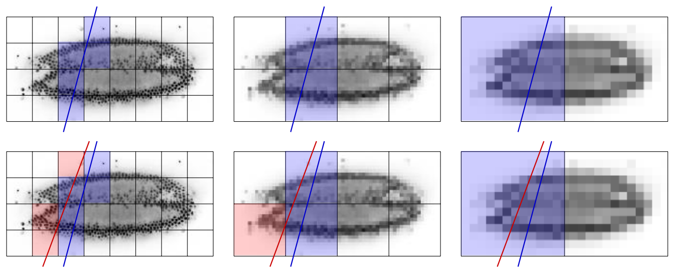


Fig. 1: BigDataViewer loading and caching scheme. (**top**) Data is stored as multi-resolution pyramid, chunked into regular blocks. To render a slice (blue line), only a subset of blocks is required. For lower resolutions, fewer blocks have to be loaded, facilitating rapid rendering and immediate user feedback. High-resolution data are loaded and filled in when the user stops browsing momentarily. (**bottom**) Recently loaded blocks are cached in RAM. For rendering the slice indicated by the red line, only the red blocks need to be loaded. Blue blocks are already cached from rendering the blue slice before.

We achieve smooth navigation of multi-terabyte image datasets by employing an intelligent loading and caching scheme, illustrated in Figure 1. To render any virtual slice, only a small fraction of the image data is relevant and needs to be loaded into memory. Our caching scheme assumes that image data is chunked into (small) regular 3D blocks and only loads blocks that are required for the current slice. Further acceleration is achieved by caching recently visited locations in memory. Moreover, the BDV makes use of multi-resolution data if available, where each image volume is stored in multiple, successively reduced resolution levels. Multi-resolution data avoids aliasing artifacts at zoomed-out views and facilitates interactive browsing. Only the most relevant scales for display are requested. Low-resolution data are loaded rapidly, providing immediate user feedback, while high-resolution detail is filled in subsequently. To facilitate this access pattern, we proposed an HDF5-based file format that is optimized for fast random access to very large data sets.

BDV is designed to be extensible and re-usable. Our file format separates metadata (in XML) and storage of voxel data (in HDF5), making it easy to adapt to other storage backends, such as various file formats or online data services. There

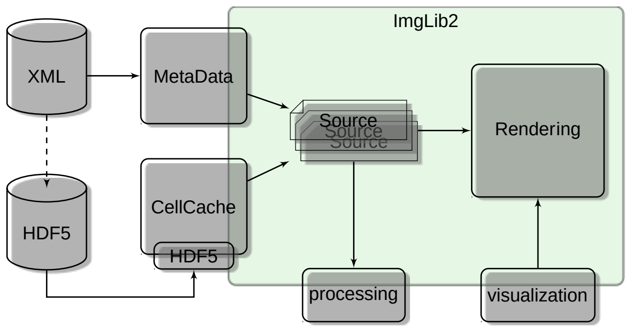


Fig. 2: The modular architecture of BigDataViewer.

already is support for the specialized file formats Imaris, KLB[2], Magellan [3], Zeiss Lightsheet Z.1, Slidebook 6[3], as well as any format that can be opened through Bioformats [4]. BDV currently provides backends for the online data servers CATMAID [5], Open Connectome [6], and DVID[4] allowing to browse massive electron microscopy (EM) datasets in previously unaccessible orientations. Additionally, we developed BigDataServer as part of BDV to make data in our own file format accessible online.

Building on the generic image processing library ImgLib2 [7], BigDataViewer has a modular architecture that separates data access backend, caching infrastructure, and visualization. This facilitates re-use of BDV components as a visualization frontend and/or data backend in image analysis and annotation tools. Section II describes BDV concepts and architecture, in particular highlighting extensibility. In Section III we discuss various applications of BDV in processing of lightsheet microscopy data, semi-automatic tracking in multiview sequences, interactive surface extraction, and curation of neuron segmentation in EM volumes.

II. Software Architecture

A. Built on ImgLib2

BigDataViewer is built on the generic image processing library ImgLib2. ImgLib2 allows clean modularization of BDV into rendering frontend and data access backend. It provides abstract interfaces between BDV modules, enabling the BDV cache backend to encapsulate and hide implementation details such as blocking and caching, exposing image volumes through a standard interface. Moreover, it facilitates rendering by lazily evaluated virtual coordinate and pixel value transformations. ImgLib2 allows to express algorithms in a way that abstracts from the data type, dimensionality, or memory storage of the image data. For BDV we rely on the following key features: virtualized pixel access, volatile pixel types, as well as transparent, virtualized image extension, interpolation, and coordinate transformations.

[2] Keller Lab Block format for lightsheet microscopy developed at Janelia Research Campus (https://bitbucket.org/fernandoamat/keller-lab-block-filetype)

[3] Intelligent Imaging Innovations format

[4] https://github.com/janelia-flyem/dvid/wiki

By virtualizing all pixel accesses, ImgLib2 decouples access to images from the storage of image data. Among the storage schemes provided by the library, the *CellImg* image container represents images by splitting them into smaller blocks (cells). We extend this functionality by loading and caching cells on demand, while still exposing the same interface as images that are completely held in memory. We use *Volatile* pixel types to represent voxels as pairs of intensity and validity. Validity in our case signals whether the intensity value exists in memory or is still enqueued to be loaded. This allows to implement a deferred loading scheme that provides immediate feedback. ImgLib2 images can be backed by transparent transformation into other images that are lazily evaluated. Whether the underlying data lives in our cache-backed images or in a standard memory array is irrelevant. This is extremely convenient for our rendering algorithm that operates under the assumption that all data is in memory all the time.

B. Re-slicing Renderer

The BDV renderer takes a set of image volumes that are registered into a common global space and displays an arbitrary slice through that global space. For each rendered pixel on the screen, the source voxels that contribute to it need to be determined. For this, volumes are transformed into a common global reference frame using their respective local-to-global transformations (registrations). Then the viewer transformation is applied to transform global coordinates into the current viewer frame. The plane $z = 0$ of the viewer frame coincides with the rendering canvas on the screen, such that voxels contributing to screen pixel (x,y) are found at coordinates $(x,y,0)$. Voxel values are then converted from their respective data type to RGB color space for display, and colors contributed by different volumes blended to a final output color. Note that all these transformations are virtualized and lazily evaluated: Only once a pixel is accessed, the transformation chain is reversed to access the corresponding source data in the cache.

For volumes that are available at multiple resolutions, each resolution level is registered individually into global space. This provides flexibility to use data sources with varying downsampling schemes. To decide which resolution level should be used for a given volume, the optimal rendering resolution is determined to best match source voxel size and on-screen pixel size. We try to always render at the optimal resolution level. However, to make optimal use of cached data, we allow resolutions to stand in for each other. Pixels currently missing in the optimal resolution level are replaced with data from other levels, while the optimal level is loading.

C. Extensible Data Format

We developed a custom open source file format that is optimized for fast random access at various scales. The file format is built on the open standards HDF5 and XML to store image volumes and metadata, respectively. Image volumes are stored as HDF5 chunked multi-dimensional arrays at successively reduced resolutions. HDF5 provides efficient input and output, supports unlimited file sizes and has built-in and extensible compression facilities.

The format is extensible in the following ways: The HDF5 file of the dataset can be replaced by alternative storage backends, of which we provide several. Moreover, the XML file of a dataset can be augmented with arbitrary additional metadata. Third-party data backends and metadata extensions are discovered automatically, using the *SciJava*[5] framework.

D. Extensible Architecture

BDV has a modular architecture that separates data access, caching, and visualization into cleanly delimited components, illustrated in Figure 2. For *Rendering* we access data through abstract *Sources*. A *Source* is a lightweight interface, providing data through standard ImgLib2 constructs. The *CellCache* triggers loading of data blocks and caches recently used blocks in RAM. Requests to load data blocks are prioritized and handled asynchronously through a pool of loader threads. Rendering is completely shielded from these implementation details. The complete dataset is exposed as ImgLib2 *CellImg* images that can be treated as if all data were in memory.

BDV can be extended with arbitrary custom *Sources*. These can be implemented on top of *CellCache* or build on entirely different mechanisms. Custom vector graphics overlays can display annotations on top of the rendered images. For external processing, it is straightforward to programmatically access the pixel data as ImgLib2 containers. Existing code for filtering and segmentation will work without modification. In the next section we discuss different applications that illustrate BDV's extension capabilities.

III. Applications

A. Multiview Reconstruction for Lightsheet Microscopy

BDV's extensible data format integrates seamlessly with Fiji's Multiview Reconstruction plug-ins for lightsheet data processing. The plug-in extends the BDV format with additional metadata (e.g., locations of segmented fluorescent beads) and data backends (e.g., Zeiss Lightsheet Z.1). It employs BDV as a frontend to allow interactive control of intermediate steps of the pipeline, see Figure 3. Individual angles of a lightsheet microscopy dataset can be viewed before and after registration [8]. Detected locations of fluorescent beads and nuclei can be visualized, and registration accuracy can be inspected in zoomed-in views. The results of a multiview deconvolution [9] and other processing steps can be incorporated into the data set and viewed in a common global space.

B. Tracking in Multiview Datasets

The temporal and spatial resolution of lightsheet microscopy combined with long time-lapses generates a torrent of data that classical annotation tools cannot handle. MaMuT[6] (Massive Multiview Tracker) is a Fiji plug-in that addresses

[5] https://github.com/scijava/scijava-common/

[6] http://fiji.sc/MaMuT

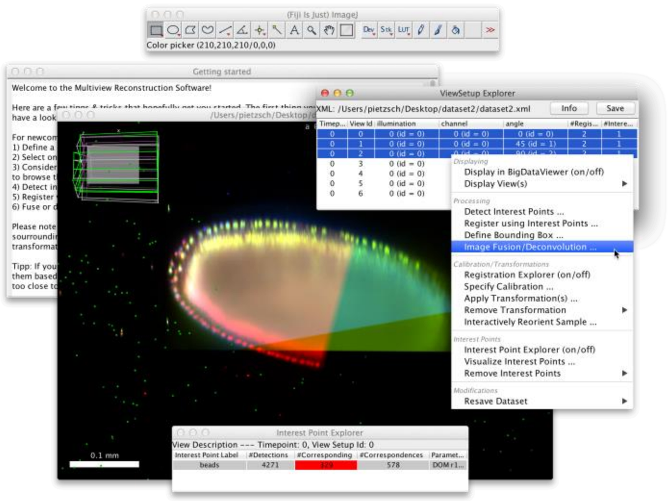

(Fiji Is Just) ImageJ
Color picker (210,210,210/0,0,0)
Getting started
Welcome to the Multiview Reconstruction Software!
ViewSetup Explorer
XML: /Users/pietzsch/Desktop/dataset2/dataset2.xml
Info
Save
Displaying
Display in BigDataViewer (on/off)
Display View(s)
Processing
Detect Interest Points ...
Register using Interest Points ...
Define Bounding Box ...
Image Fusion/Deconvolution ...
Calibration/Transformations
Registration Explorer (on/off)
Specify Calibration ...
Apply Transformation(s) ...
Remove Transformation
Interactively Reorient Sample ...
Interest Points
Interest Point Explorer (on/off)
Visualize Interest Points ...
Remove Interest Points
Modifications
Resave Dataset
0.1 mm
Interest Point Explorer
View Description --- Timepoint: 0, View Setup Id: 0

Fig. 3: The Multiview Reconstruction Fiji plug-in.

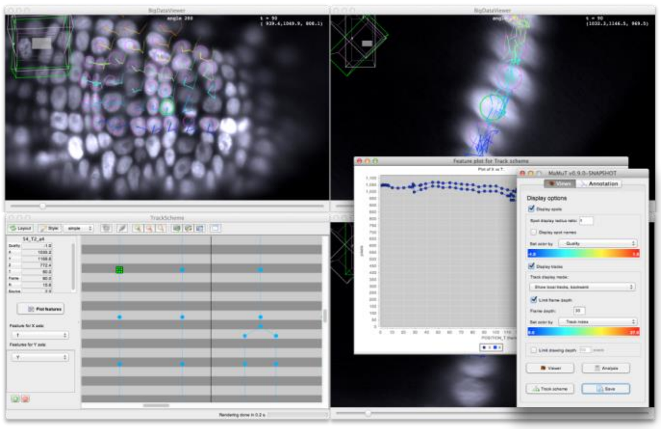


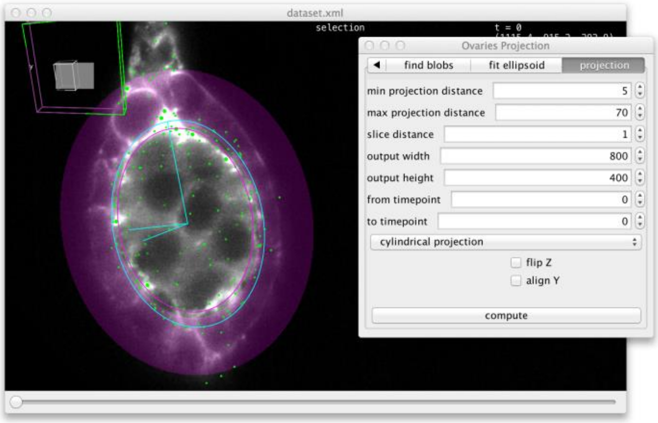
Fig. 4: The MaMuT (Massive Multiview Tracker) Fiji plug-in.

this by providing an interactive tool for the visualization, annotation, tracking and lineaging of very large, multiview image datasets. MaMuT builds on TrackMate[7], a Fiji plug-in for single particle tracking, and uses BDV both as a data backend and as a visualization frontend. It augments BDV with custom overlays for visualization of tracked cells and synchronization of multiple viewer windows. Moreover, MaMuT comprises a lineage tree "trackscheme" window, and provides simple lineage tree analysis tools. The MaMuT user interface with multiple open viewer windows is shown in Figure 4.

C. Interactive Surface Projection

BDV is well-suited as a visualization frontend and data backend for custom image processing pipelines. In Figure 5, we show a plug-in developed for semi-automatic surface extraction in time-lapses of *Drosophila melanogaster* ovaries. The objective is to project the basal side of the follicle epithelium (where planar polarized actin filaments locate) in order to track myosin dynamics. In BDV, the user roughly marks the region of interest with a bounding box.

[7] http://fiji.sc/TrackMate



min projection distance: 5
max projection distance: 70
slice distance: 1
output width: 800
output height: 400
from timepoint: 0
to timepoint: 0
cylindrical projection
☐ flip Z
☐ align Y

Fig. 5: Follicle epithelium projection tool.

In the indicated region, bright blobs are detected, which occur mainly on the apical side of the epithelium. This is followed by robustly fitting an ellipsoid surface. The user selects a projection method (cylindrical or polar) as well as a minimal and maximal distance from ellipsoid surface. The resulting projections are then computed for each timepoint, and shown as an ImageJ stack for further processing.

D. Visualization and Curation of EM Segmentations

BDV can directly access the online image services CATMAID [5], Open Connectome [6], and DVID, making massive EM datasets available in previously inaccessible orientations and complementing fixed-orientation web-browser based visualization. We are currently developing tools for visualization and manual correction of automated segmentation results. Similarly to the EM intensity data, the label fields obtained through segmentation are huge and are made accessible through online image services. In Figure 6, we show segmentation labels visualized as ARGB colors and overlaid on a FIB-SEM dataset of the adult Drosophila brain hosted by the DVID service[8]. We store label fields in a multi-resolution scheme where at coarse resolutions each voxel stores a multiset of labels, i.e., a summary of all labels the voxel covers in the full resolution data. Thanks to the abstraction provided by ImgLib2, BDV handles such non-standard data types in the same way as normal intensity values.

IV. Conclusion

The BDV is a flexible and extensible tool for the visualization, processing, and annotation of arbitrarily sized datasets. It is applicable to a variety of imaging data including lightsheet, confocal, widefield, and electron microscopy data that can be stored locally or remotely. It facilitates access to the data and annotations through an intuitive user interface, and at the same time allows programmatic access through ImgLib2 interfaces and data structures. Several tools already use and extend BDV to provide extensive reconstruction and annotation plug-ins as well as support for various data sources.

8 Data and segmentation labels were created by Janelia's FlyEM project (http://emdata.janelia.org/)

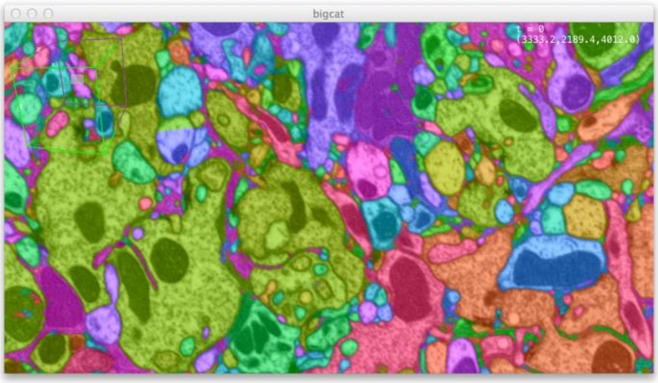


Fig. 6: Joint visualization of EM data and 3d segmentation.

Acknowledgment

We thank J. Tinevez (Institut Pasteur) for developing MaMuT; A. Pavlopoulos, I. Henry, and the Janelia FlyEM project team for providing data sets used in figures. C. Rueden and M. Hiner for developing and maintaining the ImageJ2 infrastructure on which we rely for distributing the software. S.P. was supported by the Human Frontier Science Program (HFSP) fellowship LT000783/2012, MPI-CBG, Howard Hughes Medical Institute (HHMI), and MDC. S.S. was supported by HHMI and MPI-CBG. P.T. and T.P. were supported by The European Research Council Community's Seventh Framework Program (FP7/2007-2013), grant agreement 260746.

References

[1] T. Pietzsch, S. Saalfeld, S. Preibisch, and P. Tomancak, "BigDataViewer: visualization and processing for large image data sets," *Nat Meth* **12**(6), 481–3, 2015.

[2] J. Schindelin et al., "Fiji: an open-source platform for biological-image analysis," *Nat Meth* **9**(6), 676–82, 2012.

[3] H. Pinkard et al., "µMagellan: A flexible, open source platform for high-level automation of high throughput biological light microscopy," in preparation.

[4] M. Linkert et al., "Metadata matters: access to image data in the real world," *JCB*, **189**(5), 777–82, 2010.

[5] S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomancak, "CATMAID: Collaborative annotation toolkit for massive amounts of image data," *Bioinformatics* **25**(15), 1984–6, 2009.

[6] R. Burns et al., "The Open Connectome Project Data Cluster: Scalable Analysis and Vision for High-Throughput Neuroscience," *in SSDBM* **25**, art. 27, 2013.

[7] T. Pietzsch, S. Preibisch, P. Tomancak, and S. Saalfeld, "ImgLib2—generic image processing in Java," *Bioinformatics* **28**(22), 3009–11, 2012.

[8] S. Preibisch, S. Saalfeld, J. Schindelin, and P. Tomancak, "Software for bead-based registration of selective plane illumination microscopy data," *Nat Meth* **7**(6), 418-9, 2010.

[9] S. Preibisch, et. al, "Efficient Bayesian-based multiview deconvolution," *Nat Meth* **11(**6), 645–8, 2014