

# Efficient processing and analysis of large-scale light-sheet microscopy data

Fernando Amat, Burkhard Höckendorf, Yinan Wan, William C Lemon, Katie McDole & Philipp J Keller

Howard Hughes Medical Institute, Janelia Research Campus, Ashburn, Virginia, USA. Correspondence should be addressed to P.J.K. ([kellerp@janelia.hhmi.org](mailto:kellerp@janelia.hhmi.org)) or F.A. ([amatf@janelia.hhmi.org](mailto:amatf@janelia.hhmi.org)).

Published online 1 October 2015; doi:10.1038/nprot.2015.111

**Light-sheet microscopy is a powerful method for imaging the development and function of complex biological systems at high spatiotemporal resolution and over long time scales. Such experiments typically generate terabytes of multidimensional image data, and thus they demand efficient computational solutions for data management, processing and analysis. We present protocols and software to tackle these steps, focusing on the imaging-based study of animal development. Our protocols facilitate (i) high-speed lossless data compression and content-based multiview image fusion optimized for multicore CPU architectures, reducing image data size 30–500-fold; (ii) automated large-scale cell tracking and segmentation; and (iii) visualization, editing and annotation of multiterabyte image data and cell-lineage reconstructions with tens of millions of data points. These software modules are open source. They provide high data throughput using a single computer workstation and are readily applicable to a wide spectrum of biological model systems.**

## INTRODUCTION

Light-sheet microscopy is an optical sectioning technique<sup>1–3</sup> that provides high imaging speed and high spatial resolution over long periods of time, while minimizing energy load on the biological system under observation<sup>4–15</sup>. Owing to these powerful capabilities, light-sheet microscopy has emerged as a key method for live imaging in cell biology and developmental biology<sup>16–20</sup>, as well as in neuroscience<sup>21–23</sup>. By capturing fast developmental and functional processes at the single-cell level across entire, complex biological systems, light sheet–based imaging can address fundamental biological questions that are not accessible with previous methods. In the domain of developmental biology, it has become feasible to systematically follow populations of progenitor cells as they form tissues, organs and even entire embryos. Such system-level cell-lineage reconstructions provide important insights into the stereotypy of developmental processes, link developmental history to cell function in the developmental building plan of an animal, aid in dissecting the role of differential gene expression in directing cell-fate decisions, and facilitate experimental validation of mechanistic models of development<sup>24–33</sup>. In neuroscience, light-sheet microscopy has made it possible to perform functional imaging of large neuronal populations, entire brains<sup>5,34</sup> and even the entire CNS<sup>35</sup>. Such experiments have the potential to illuminate how large neural networks perform complex computations and generate behavior at the single-cell level<sup>22,36</sup>.

However, light-sheet imaging experiments produce vast amounts of complex image data; from long-term imaging of developing embryos to high-speed functional imaging of the brain, each light-sheet recording consists of up to several tens of terabytes of multidimensional image data (including three spatial dimensions, time and multiple color channels). Thus, data management, as well as image processing and data analysis, rather than the experiments themselves, can easily become the bottleneck on the path to biological discovery. A computational framework that addresses these challenges, and does so with high data throughput and at minimal cost to the investigator, is crucial for routinely recording light-sheet data sets and for extracting biologically relevant information.

Here we present detailed protocols for operating a computational pipeline that efficiently handles the spectrum of challenges encountered with light-sheet microscopy image data, from high-throughput lossless data compression to content-based multiview image fusion. We furthermore provide protocols and software for large-scale cell tracking in developmental image data sets, as well as for large-scale image data visualization and annotation.

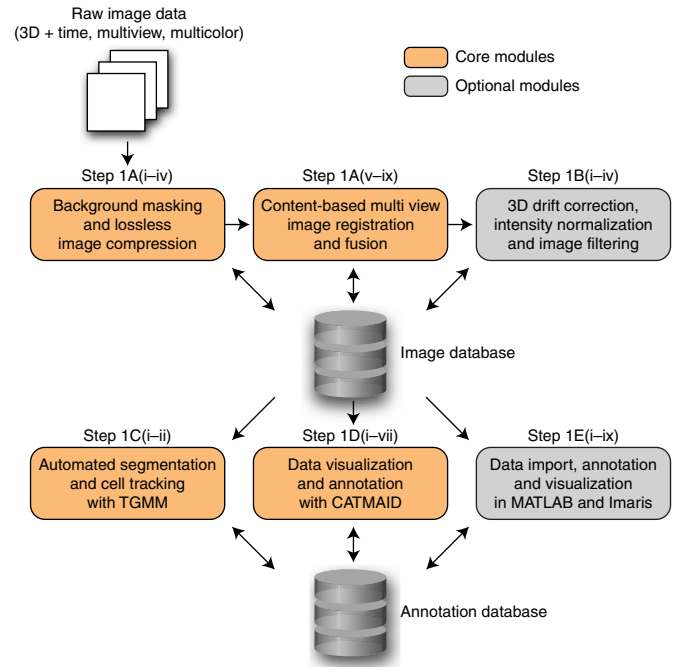
## Development of the protocol

In the protocols presented here, we describe five main computational modules (**Figs. 1 and 2; Supplementary Software 1–6**): first, our block-based lossless compression file format for efficiently storing large amounts of image data and rapidly retrieving arbitrary regions of interest; second, MATLAB scripts for content-based registration and fusion of time-lapse, multiview image data; third, our Tracking with Gaussian Mixture Models (TGMM) software for automated large-scale segmentation and tracking of fluorescently labeled cell nuclei; fourth, a branch of the Collaborative Annotation Toolkit for Massive Amounts of Image Data (CATMAID)<sup>37,38</sup> for visualizing 5D microscopy data sets and editing associated cell tracking results; and fifth, MATLAB scripts for importing, analyzing and visualizing large-scale cell-lineage reconstructions.

All protocols have been extensively tested on long-term *in vivo* time-lapse recordings of multicellular organisms, such as fruit fly, zebrafish and mouse embryos, primarily using data generated with SiMView light-sheet microscopy<sup>8,39</sup>. In addition, our processing pipeline has been successfully applied to other microscopy modalities, such as confocal fluorescence microscopes and commercial light-sheet microscopes<sup>39</sup>, and other model systems, such as *Parhyale* and *Platynereis* embryos<sup>40</sup>, as well as fruit fly and zebrafish larvae<sup>5</sup>. Our framework tackles various large-scale image processing challenges, including the analysis of multiterabyte developmental image data sets for system-level cell tracking (with tens of millions of tracked cell locations per embryo)<sup>39</sup> and management of multiterabyte functional image data sets produced by whole-brain<sup>5</sup> or whole-CNS<sup>35</sup> calcium imaging.

**Figure 1** | Overview of image processing and data analysis workflow. The computational framework described in our protocols addresses typical data management, image processing and data analysis challenges encountered in light-sheet microscopy experiments. Starting with the raw image data sets, which consist of up to several terabytes of 3D images recorded as a function of time and comprise up to several color channels and view angles, the computational modules described here facilitate rapid data compaction via adaptive image background and foreground detection, background masking and image compression in our lossless KLB file format optimized for large-scale image data and multicore CPU architectures; high-throughput content-based multiview image registration and fusion for SiMView-like multiview data sets comprising up to four orthogonal views; 3D drift correction, intensity normalization and adaptive background correction; automated segmentation and cell tracking using our software framework TGMM; large-scale image data visualization and editing of cell-lineage reconstructions using a branch of CATMAID for 5D light microscopy image data sets; and data import/export between TGMM, CATMAID and the commercial rendering software Imaris. All of these software modules can be used individually or as part of our integrated computational pipeline.

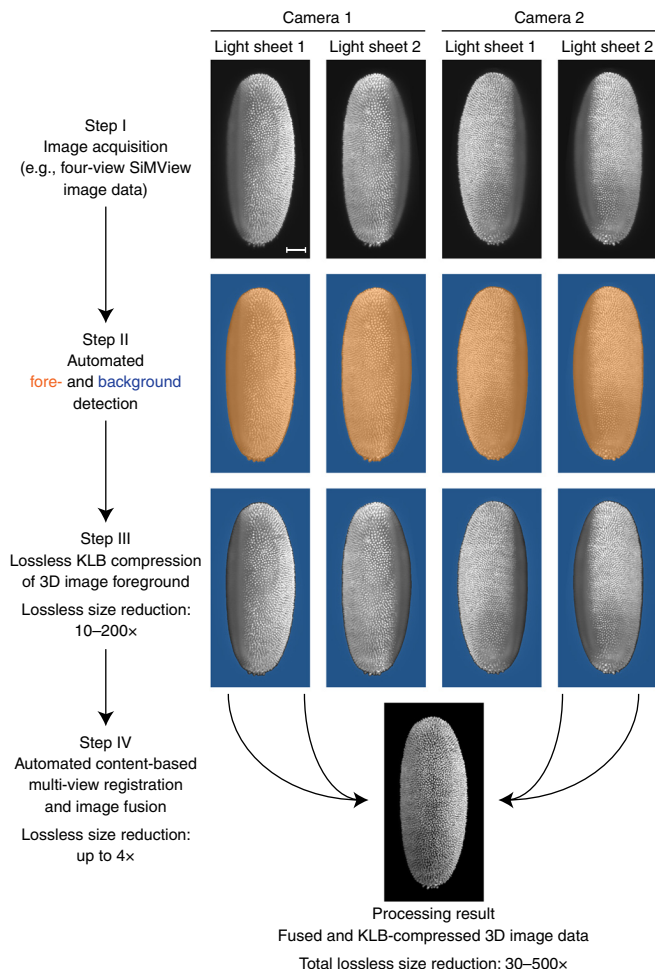
Moreover, our modules for image compression, multiview fusion, segmentation and cell tracking are also suitable for applications that require real-time performance; i.e., our pipeline is capable of processing speeds exceeding the data acquisition rate of the light-sheet microscope, using a single computer workstation equipped with a conventional compute unified device architecture (CUDA)-enabled graphics card.



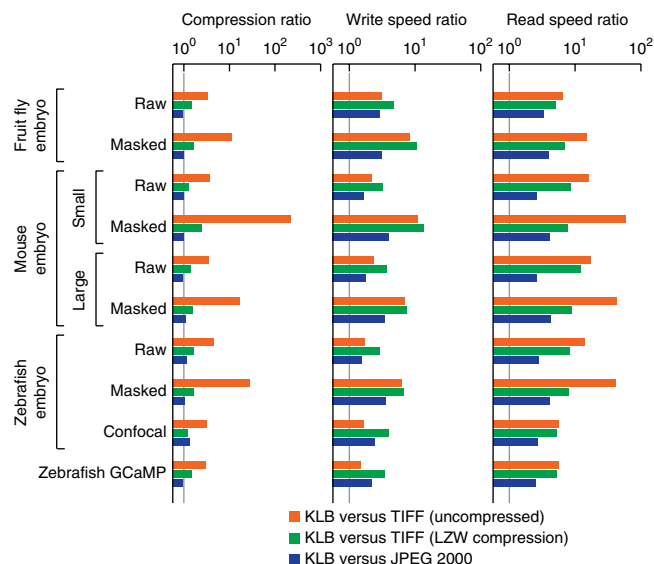
## Comparison with other methods

One of the key challenges in developing computational tools for light-sheet microscopy image data is scalability. There is a vast amount of literature and software related to the computational problems discussed here, such as data compression, visualization, registration, segmentation and tracking. However, many of these existing approaches either break down or are too time-consuming and resource-intensive when applied to multiterabyte image data sets. In this section, we compare our computational modules with existing methods that have been tested in similar data sets in terms of image characteristics and (if applicable) size.

**Figure 2** | Lossless image compression and content-based multiview fusion. The first set of modules in our computational framework for high-throughput image processing is designed for rapid lossless data compaction of single-view or multiview light-sheet microscopy data sets. Step I: acquisition of light-sheet microscopy image data. These raw images are used as input data in the next step. Step II: automated detection of image foreground and background; i.e., detection of image regions that correspond to parts of the specimen (foreground, shown in yellow) or to regions that are either outside the specimen or do not contribute fluorescent signal (background, shown in blue). Step III: masking of image background (i.e., populating the automatically detected background regions, which contain only background noise, with zeros) and lossless compression of the image data using the KLB file format. Step IV: automated content-based multiview image registration and image fusion. Steps II and III are applicable to both single-view and multiview image data sets, whereas Step IV is designed for high-throughput image fusion of SiMView-like multiview data sets comprising up to four orthogonal views. These image processing steps combine the high-quality image information of all recorded views into a single, information-rich image stack of the entire specimen, and efficiently store the 3D image data in a lossless image format. Thereby, the pipeline markedly reduces the size of the raw image data (on average by a factor of 180, **Fig. 4**) without discarding or changing parts of the original 3D image data that contain potentially useful information. At the same time, the pipeline is real-time capable; that is, all processing steps are completed in less time than required for image acquisition itself (**Table 1**). Data compaction performance numbers are based on the fruit fly, mouse and zebrafish imaging experiments shown in **Figure 4**. Scale bar, 50  $\mu\text{m}$ .



**Figure 3** | Performance comparison of lossless image compression formats. Performance of the KLB lossless compression format versus LZW-TIFF (green) and JPEG 2000 (blue) lossless compression formats with respect to compression ratio (first column), write speed (second column) and read speed (third column). The JPEG 2000 benchmark uses the multithreaded commercial library PICTools Medical SDK (Accusoft). A performance comparison of KLB and uncompressed TIFF formats is included as well (orange). LZW-TIFF and uncompressed TIFF benchmarks use the 'imread' and 'imwrite' functions provided by the Image Processing Toolbox in MATLAB. All performance data are provided as ratios with KLB performance in the numerator; i.e., ratios larger than one (gray lines) indicate superior performance of the KLB file format. The comparison was performed using a variety of fluorescence microscopy image data sets located on a high-performance network-attached storage server connected to the image processing workstation via 10 Gb s<sup>-1</sup> glass fiber. Benchmark data sets include SiMView light-sheet microscopy recordings of fruit fly, mouse and zebrafish embryonic development (data sets 1–8), confocal microscopy data of a zebrafish embryo (data set 9) and SiMView functional image data of brain activity in a larval zebrafish (data set 10). Developmental data sets (data sets 1–8) were analyzed as raw and masked versions in order to illustrate the importance of background masking for maximizing data storage and to access efficiency. Please see steps I–III in **Figure 2** for a description of the concepts underlying background masking. Note that read speeds for uncompressed TIFF files are particularly low, as a large fraction of time is spent on accessing the large files. If image data sets are small enough for a local storage solution—i.e., when using the same computer for long-term data storage and image processing—the data access time overhead encountered for uncompressed image data can be slightly reduced, e.g., through the use of a high-performance RAID array. For benchmarks performed with image data sets stored locally on a high-performance RAID array built from solid-state drives (SSDs), please see **Supplementary Figure 1**. For information about the block-size dependency of KLB performance, please see **Supplementary Figure 2**.



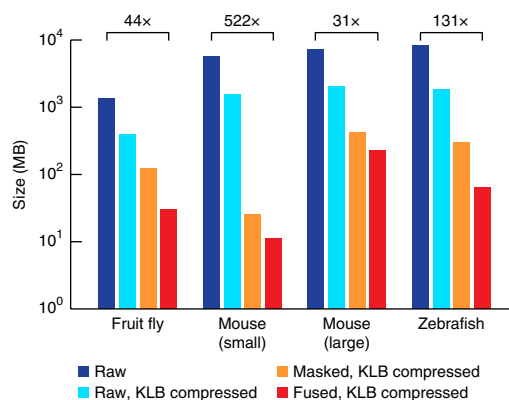
**Image compression.** In our comparison of image-compression formats, we focus on formats that have found widespread use and that offer lossless compression capability, as researchers usually want to store an unaltered version of their data. JPEG 2000 is one of the most widely used image compression formats. However, although the JPEG 2000 standard provides a description of 3D compression, few implementations of this capability actually exist. Most software packages compress image data plane by plane, which is inefficient for retrieving arbitrary regions of interest in large multidimensional image volumes. Moreover, it is difficult to efficiently parallelize all JPEG 2000 coding and decoding steps, which makes it challenging to take full advantage of modern multicore computing hardware.

HDF5 is another popular container for image files. Aside from offering lossless data compression, HDF5 is capable of storing data in blocks for fast retrieval of arbitrary regions of interest. Unfortunately, the HDF5 interface does not parallelize writing operations, which negatively affects speed.

To overcome these limitations, we developed the Keller Lab Block (KLB) lossless image-compression format, which combines high compression ratios, fast read/write speeds and a flexible block architecture that enables efficient access to arbitrary regions of interest (**Figs. 3 and 4; Supplementary Figs. 1–3**). Inspired by Parallel BZip2, a common Linux compression module, we partition images in 5D blocks and compress all blocks in parallel using BZip2. Both reading and writing operations are parallelized, and they scale linearly with the number of cores in the CPU (**Fig. 5**). In addition, we provide a simple API for interfacing the open-source C++ code with various platforms, as well as an interface file for the SWIG tool, which can be used to autogenerate wrapper code for various languages, including Java, C#, Python, Perl and R (**Supplementary Software 1**).

By using a variety of fluorescence microscopy data sets, we compared KLB performance with that of other state-of-the-art compression formats (**Fig. 3 and Supplementary Figs. 1–3**), including one of the most efficient multithreaded implementations

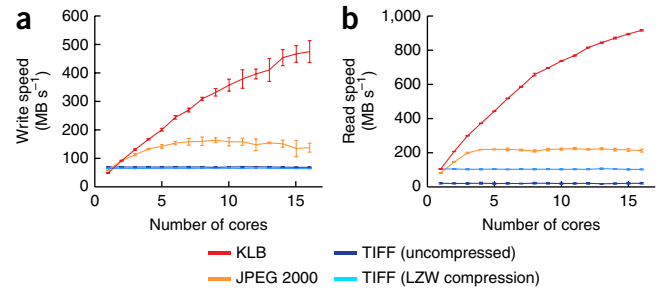
**Figure 4** | Multiview image data compaction for light-sheet microscopy. Comparison of image file sizes obtained by taking advantage of our pipeline for image data compaction (**Fig. 2**) to varying degrees. Data set sizes are shown for raw, uncompressed image data sets (dark blue, step I III in **Fig. 2**), for KLB-compressed raw data sets (light blue), background-masked, KLB-compressed data sets (orange, steps I–III in **Fig. 2**) and for multiview fused, background-masked, KLB-compressed data sets (red, steps I–IV in **Fig. 2**). Even when recording only single views of a specimen, i.e., if multiview image fusion is not applicable, background masking and lossless KLB compression alone already lead to a substantial reduction in data size, without loss of information. The four types of image data sets included in this comparison represent single time points of time-lapse recordings of fruit fly, mouse and zebrafish embryos acquired with SiMView light-sheet microscopy. The factors shown above each set of bars indicate total data set size reduction from raw, uncompressed multiview data format to fused, background-masked, KLB-compressed data format. Note that data set sizes shown in this figure represent image size per time point and thus scale linearly to large-scale light-sheet microscopy time-lapse recordings comprising thousands of time points and tens of terabytes of image data.



**Figure 5** | Image compression performance using multicore CPUs.

(a,b) Write (a) and read (b) speeds as a function of available CPU cores, for the uncompressed TIFF file format (dark blue), as well as lossless KLB (red), JPEG 2000 (orange) and LZW-TIFF (light blue) file formats. The benchmark was performed using data set 6 in **Figure 3**. Note that uncompressed and LZW-compressed TIFF file formats do not benefit from multicore CPU architectures. JPEG 2000 can partially leverage the processing power of a small number of CPU cores (no performance increase observed beyond 4 CPU cores). In contrast, KLB performance scales almost linearly with the number of CPU cores, even when using multicore processing architectures with as many as 16 CPU cores.

The JPEG 2000 benchmark uses the multithreaded commercial library PICTools Medical SDK (Accusoft). LZW-TIFF and uncompressed TIFF benchmarks use the 'imread' and 'imwrite' functions provided by the Image Processing Toolbox in MATLAB. Error bars represent s.d. for  $n = 5$  iterations of the benchmark. For information about the block-size dependency of KLB performance, please see **Supplementary Figure 2**.



of JPEG 2000 (PICTools Medical SDK, Accusoft). When KLB is used for locally stored image data (**Supplementary Fig. 1**), it provides superior compression ratios (3% and 70% better than JPEG 2000 or LZW-compressed TIFF, respectively) and read/write speeds (3.2-fold and 4.5-fold faster than JPEG 2000 or LZW-compressed TIFF, respectively, using 16-CPU cores). When KLB is used for network-attached image data (the typical setting for large-scale image data sets, **Fig. 3**), improvements in speed are even higher (3.3-fold and 7.5-fold faster than JPEG 2000 or LZW-compressed TIFF, respectively, using 16-CPU cores). Compared with uncompressed TIFF format, KLB provides markedly improved read/write speeds (3.1-fold and 16.5-fold faster locally or over the network, respectively), which is a direct result of the rapid data compaction in KLB and the reduced transfer times for compressed image data. Thus, KLB outperforms state-of-the-art file formats with respect to both compression ratio and speed by taking full advantage of modern multicore CPUs, and it offers lossless data compaction of large-scale image data sets with minimal access latency.

**Multiview image fusion.** An efficient cross-platform multiview image fusion method using embedded fluorescent beads surrounding the sample has been incorporated in Fiji as part of the 'Multiview Reconstruction' plug-ins<sup>41</sup>. This bead-based method allows registration of any number of views distributed in an arbitrary geometry, without prior information about the relative location of each view. As a generalization of its initial design for bead-based registration, the method has more recently been extended to support image data containing other types of blob-like features (such as fluorescent cell nuclei) that can be reliably detected with a Difference of Gaussians filter.

In contrast, the multiview fusion module provided by our processing pipeline (**Supplementary Software 3**) is complementary in several ways. Our module does not require and rely on specific features to facilitate registration, but rather it uses all image information present in the sample itself, irrespective of the type of fluorescent label used in the experiment. Fast content-based registration is achieved by introducing the assumption of a multiview imaging assay with up to four orthogonal views (using up to two opposing light sheets and two opposing cameras); i.e., our method is not capable of registering arbitrary views. This latter constraint represents the main limitation of our method. However, as a direct result of this design principle, our method does not require the presence of fluorescent blob-like structures in the sample to facilitate accurate registration

and image fusion. This approach thus offers the following three advantages: (i) our method is applicable to large specimens and high-magnification imaging experiments, for which the field of view does not cover space outside the volume of the biological specimen itself (and hence lacks space for beads); (ii) our method provides flexibility for biological sample preparation, as it does not require the sample to be embedded in an agarose gel or a similar matrix suitable for anchoring beads; and (iii) our method can partially compensate for the effect of light refraction along the light path through the sample, as our alignment is based on image information inside the sample. Our method is furthermore designed for high-throughput image processing (**Tables 1 and 2**), and it offers real-time capability for large-scale light-sheet microscopy data sets: by using a single computer workstation, our registration and fusion pipeline generally processes image data at a rate faster than the data acquisition rate of the light-sheet microscope<sup>39</sup> (**Table 1**).

**Image segmentation and cell tracking.** There are several freely available computational methods for nuclei segmentation and cell tracking. These methods were specifically developed for cell-lineage reconstructions using time-lapse light microscopy images of fluorescently labeled nuclei. However, most of these approaches have been developed for relatively small model organisms, such as *Caenorhabditis elegans* embryos<sup>42–44</sup>, which undergo stereotyped development and comprise several hundred cells by the end of embryonic development, or for very early developmental stages of more complex multicellular organisms, such as the early zebrafish blastula<sup>45,46</sup> and the *Drosophila* blastoderm<sup>8,46</sup>. These methods do not aim to facilitate automated cell lineaging in later stages of development, and their underlying design principles either produce high error rates in such data sets or do not scale to the tens of thousands of cells encountered during advanced embryogenesis of vertebrates and higher invertebrates<sup>39</sup>. An accurate method that scales to large data sets is available for cell nuclei segmentation<sup>47</sup>, although this method does not perform cell tracking. Only very recently have existing methods<sup>48</sup> for joint segmentation and tracking been successfully extended to handle data recorded in later developmental stages, although scalability with increasing cell counts is still an issue. In contrast, computation time of the TGMM software included in our framework (**Supplementary Software 4**) scales linearly with the number of segmented and tracked objects while maintaining state-of-the-art accuracy even in late developmental stages: on a single computer workstation equipped with a Tesla K20 graphics processing



**TABLE 1** | Computation time requirements of image processing pipeline.

Computational module	Computation time (s)	Computation time per time point (s)
<b>clusterPT.m</b>		
• sCMOS image correction	8.36 per time point	8.36
• Background masking		
• KLB lossless compression		
<b>clusterMF.m</b>		
• Multiview registration	19.49 per ten time points	1.95
• Multiview image fusion		
<b>localAP.m</b>		
• Parameter interpolation	5.09 per experiment	0.04
<b>clusterTF.m</b>		
• Multiview image fusion	7.88 per time point	7.88
<b>ProcessStack</b>		
• Hierarchical segmentation	2.73 per time point	2.73
<b>TGMM</b>		
• Cell tracking	8.29 per time point	8.29
• Detection of cell divisions		
• Filtering of cell lineages		

The table shows computation time requirements of each module of the image processing pipeline, from image correction, masking and lossless compression of the raw image data with clusterPT.m (Step 1A(i)) to cell tracking and reconstruction of cell lineages with TGMM (Step 1C(ii)). All measurements were performed using adaptive blending for image fusion. The benchmarks are based on the processing of 120 time points of a typical SiMView four-view light-sheet microscopy experiment capturing the development of an entire *Drosophila* embryo. The four-view image data were recorded in 30-s intervals; that is, the test data set represents one hour of live imaging. Image processing up to final multiview image fusion (clusterPT.m, clusterMF.m, localAP.m, clusterTF.m) took 18.23 s per time point and is thus almost twice as fast as the image acquisition process itself. Segmentation, cell tracking and reconstruction of cell lineages (ProcessStack, TGMM) took 11.02 s per time point, including all read/write operations. Thus, the total computation time per time point (29.25 s) is shorter than the time point interval in the image acquisition process.

unit (GPU), processing speed is on average 26,000 cells per min, which enables real-time performance in all tested scenarios<sup>39</sup>. The software is designed for easy use without prior domain knowledge, and it requires adjustment of only two framework parameters when applied across multiple model systems and imaging modalities. We note that the most important factor that influences tracking accuracy is the temporal sampling of cell movements in the time-lapse data, although image quality and cell density can affect results as well<sup>39</sup>.

**Data visualization and editing of cell-lineage annotations.** OMERO<sup>49</sup> is a software solution that is exceptional in its data organization features. OMERO facilitates organizing, remote browsing and analysis of multidimensional microscopy data. It excels at providing unified access to images and metadata from multiple sources and a plethora of file formats in a multiuser environment. As such, it supports specialized applications that are beyond its own scope. Newer versions of OMERO store data in their original files; this strategy is guaranteed to be lossless, but it is reliant on third-party choices of data file layout and compression algorithms, which are crucial parameters when balancing storage efficiency and interactive visualization.

Multiple software options provide the ability to concurrently visualize image data and edit cell-lineage reconstructions. goFigure2 is an open-source cross-platform software<sup>50</sup> specifically designed for this task. Similarly to CATMAID, it uses a database to store all segmentation and tracking information, which allows it to efficiently handle millions of data points and to import results into other modules for downstream analysis. goFigure2 uses the VTK library<sup>51</sup> for visualization and 3D rendering, which provides more visualization options than CATMAID. However, as images are not partitioned in small chunks of data (tiles) ahead of time, navigating the data along the time axis of a time-lapse imaging experiment requires constantly loading image stacks from disk. This requirement precludes real-time interaction with large image data sets. Imaris (Bitplane) is a commercial scientific software for data visualization, segmentation and analysis of 3D and 4D microscopy data sets, and it includes a module for cell tracking. Like goFigure2, Imaris offers 3D rendering options for advanced data visualization and, if a sufficient amount of GPU memory is available, consecutive time points are cached for smooth transition between time points in a short temporal window. However, all data (images, segmentation and tracking annotations) associated with a given project are stored in a single HDF5-like file, which appears to substantially slow performance when using multiterabyte image data sets and millions of tracked data points. Moreover, neither goFigure2 nor Imaris allows concurrent remote data access by multiple users; this capability is particularly valuable for large-scale collaborative projects that involve multiple entities around the globe.

These limitations are addressed in CATMAID<sup>37,38</sup>, which allows rapid, uninterrupted browsing of multiterabyte data sets and concurrent large-scale data annotation involving tens of millions of data points, even when accessing the data remotely through the internet (Fig. 6). Our branch of the CATMAID framework (**Supplementary Software 5**) currently supports light microscopy image data sets with up to five dimensions (three spatial dimensions, time and color).

Alternative software solutions for visualizing large-scale (i.e., larger than locally available memory) 5D data sets on single computer workstations are increasingly becoming available, and they include both commercial and open-source software, such as Arivis Vision 4D, Amira, Vaa3D (refs. 52,53) and BigDataViewer<sup>54</sup>. Each of these software packages includes different visualization tools, although most of them follow similar principles, such as the use of multiscale block-based file formats for efficient data access in regions of interest at the appropriate level of resolution. Some of these software solutions furthermore already include or are starting to incorporate editing and annotation tools on top of their visualization engines.

## Experimental design

All software modules are available from <http://www.janelia.org/lab/keller-lab> and as **Supplementary Software 1–6**, and they have been tested on multiple operating systems (including Windows, Linux and Mac OS), except for the backend required by the web application CATMAID, which has only been tested on a Linux platform. However, CATMAID can, in principle, also be set up on other operating systems. We provide source code and documentation for all modules to enable their adaption to specific needs and various types of imaging experiments. Although all five modules can be used independently, they are



PROTOCOL

TABLE 2 | Memory requirements of image processing pipeline.

Computational module	Module configuration <sup>a</sup>	Estimated memory consumption <sup>b</sup>
<b>clusterPT.m</b>		
• sCMOS image correction	rotationFlag = 0	$1.2 \times (2n + 2) \times S$
• Background masking	rotationFlag $\neq$ 0	Up to $1.2 \times (2n + 4) \times S$
• KLB lossless compression		
<b>clusterMF.m</b>		
• Multiview registration	Wavelet fusion, 4 views	$13.2 \times S$
• Multiview image fusion	Wavelet fusion, 2 views	Up to $10.8 \times S$
	Other fusion, 4 views	$9.6 \times S$
	Other fusion, 2 views	Up to $8.4 \times S$
<b>clusterTF.m</b>		
• Multiview image fusion	Wavelet fusion, 4 views	$9.6 \times S$
	Wavelet fusion, 2 views	$7.2 \times S$
	Other fusion, 4 views	$6.0 \times S$
	Other fusion, 2 views	$3.6 \times S$
<b>clusterCS.m</b>		
• 3D drift correction	All settings	$5.5 \times S$
• Intensity normalization		
<b>clusterFR.m</b>		
• Local background correction	All settings	$3.6 \times S$

The table shows conservative estimates of memory consumption of various core modules of the image processing pipeline. The estimate considers all major computations and an additional buffer of 20% to account for minor computations. <sup>a</sup>The configuration setting 'other fusion' refers to the use of adaptive blending, geometrical blending or averaging in the modules clusterMF.m or clusterTF.m (parameter 'fusionType'). <sup>b</sup>The formulas for estimated memory consumption include two parameters, one specific to clusterPT.m (parameter 'n') and one that applies to all modules (parameter 'S'). Parameter 'n' is the maximum number of image channels that are combined to build segmentation masks in clusterPT.m; i.e., it is equal to the number of columns of the matrix 'references' if this matrix is not empty, or equal to 1 if the matrix 'references' is empty. Parameter S is the size of a single-view, single-channel image stack at a single time point, assuming that image data are stored in uint16 format (i.e., S is equal to the number of voxels in the image stack times two bytes).

also capable of communicating results to each other and form an integrated processing pipeline. It is furthermore possible to integrate the respective functionality of each module in other software packages (for example, we offer full ImageJ/Fiji support for our block-based image file format). Finally, all modules can be run efficiently on a single computer workstation equipped with MATLAB (MathWorks) and a CUDA-enabled graphics card, and most of our modules are capable of taking full advantage of modern multicore CPUs and GPUs, as well as cluster environments.

Applications of the protocol

The methods described here can be applied to image data from a variety of imaging techniques<sup>39</sup>, including custom-built light-sheet microscopes, commercial light-sheet microscopes and confocal fluorescence microscopes. In our laboratory, we are routinely using this set of computational tools for image data management and processing of SiMView<sup>5,8</sup> and hs-SiMView<sup>35</sup> light-sheet microscopy image data sets spanning a range of biological model systems, including zebrafish embryos and larvae, *Drosophila* embryos, larvae, pupae and adults, mouse embryos, *Platynereis* embryos and *Parhyale* embryos. This list can, in principle, be extended to any biological specimen suitable for imaging with

optical sectioning fluorescence microscopy in general and light-sheet microscopy in particular. Specific examples of previous use cases in systems neuroscience include data management of large-scale functional imaging data of the zebrafish larval brain<sup>5</sup> and the CNS of larval *Drosophila*<sup>35</sup>, which were acquired using state-of-the-art calcium indicators GCaMP5G (ref. 55) and GCaMP6s (ref. 56), respectively. In the field of developmental biology, the methods presented here have previously been used for data management, multiview fusion, whole-embryo long-term cell tracking, as well as data curation and visualization in zebrafish, *Drosophila*, mouse and *Platynereis* embryos<sup>8,39,40,57</sup>. For cell tracking and cell lineaging applications, such as our cell-lineage reconstruction of the early *Drosophila* nervous system, our tools are typically most effective for image data of organisms ubiquitously expressing nuclei-localized fluorescent markers. In the following paragraphs, we provide information about application details specific to individual modules of the processing pipeline.

We note that, although our content-based multiview fusion module does not support arbitrary optical geometries, it is compatible with some of the most commonly encountered light-sheet microscope configurations. Aside from the SiMView four-view geometry (providing up to four camera/light-sheet view combinations through the use of two detection arms and two light sheets whose optical axes are arranged as a cross), it is also possible

to process data from multiview setups that rely on mechanical rotation by 180° to acquire complementary views of the specimen, as well as from bidirectional illumination setups that use two light sheets along the same illumination axis. Such configurations include OpenSPIM setups<sup>58,59</sup>, as well as commercial light-sheet microscopes—e.g., the Lightsheet Z.1 by Carl Zeiss.

Our TGMM software can generally be used to track blob-like structures in various types of 2D or 3D time-lapse images, as long as object movements between consecutive time points do not exceed object size. CATMAID is capable of visualizing arbitrary 5D image data, and it allows generating and editing object annotations that can be naturally organized in tree-like structures, thus encompassing essentially any type of segmentation and tracking task. CATMAID was initially developed for visualizing and annotating large electron microscopy data sets generated in the field of connectomics for reconstructing the wiring diagram of the brain at nanometer resolution<sup>60</sup>. This software is thus also well suited to microscopy data of neural tissues from light-based imaging modalities<sup>61–63</sup>.

Finally, our KLB compression algorithm can be applied to any type of image data (consisting of signed or unsigned integers with a depth of 8, 16, 32 or 64 bits, as well as 32-bit or 64-bit floating

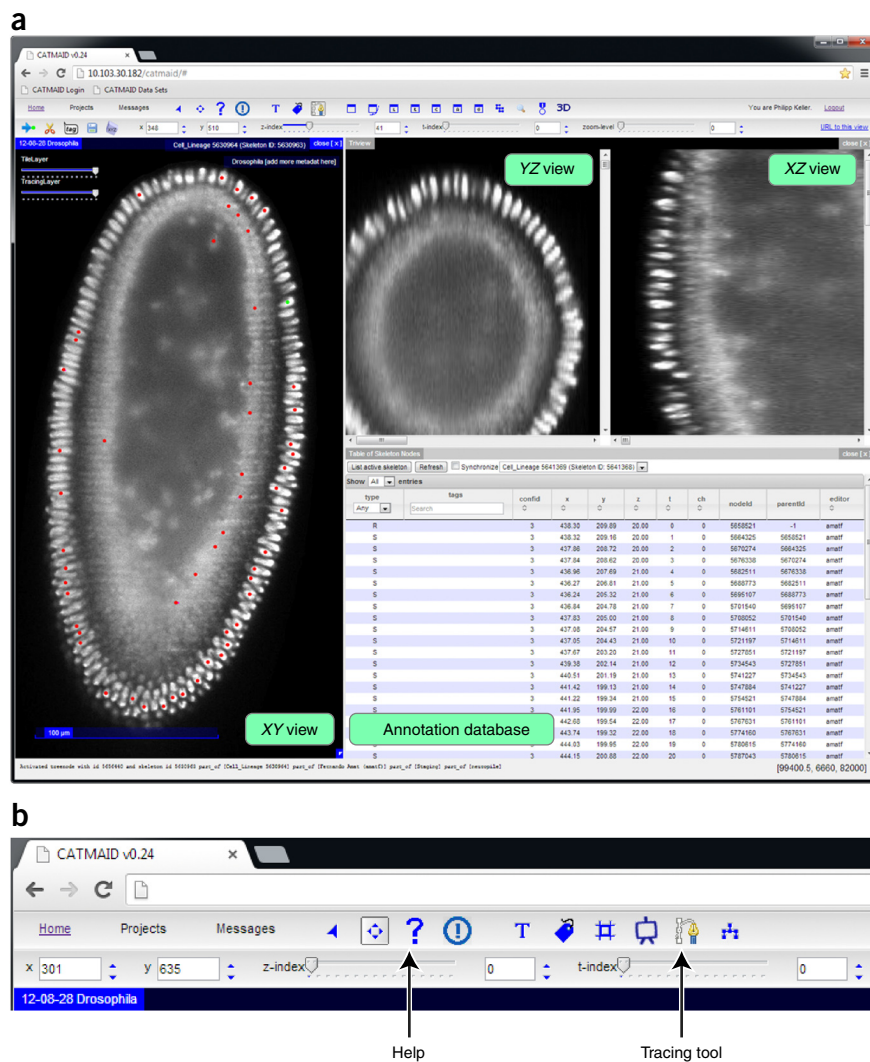
**Figure 6** | Image annotation and editing of cell-lineage data using CATMAID. (a) Screenshot of internet browser showing CATMAID GUI during the manual curation of TGMM cell-lineage data in a fruit fly embryo. Image data are displayed superimposed with cell-lineage data points in a tri-view arrangement (XY, YZ and XZ slices of the specimen). Both image data and cell-lineage annotations are stored remotely on a server to avoid data duplication; that is, the same image data set can be used for multiple cell lineaging projects. The annotation database containing the full cell-lineage reconstruction is shown in the bottom right corner. (b) Enlarged view of a part of the CATMAID toolbar, which provides utilities for browsing the image data, as well as accessing and editing data annotations.

point data with up to five dimensions), irrespective of its source. In principle, any type of microscopy data benefit from the file size reduction and high read and write speeds achieved by KLB. The block-based design of KLB is furthermore particularly helpful when working with large image volumes, such as image data of entire developing embryos<sup>39</sup>, as well as large neural tissues or entire brains treated with chemical clearing methods<sup>61–63</sup>, as the KLB format provides rapid access to local image regions with minimal overhead.

### Level of expertise needed to implement the protocol

Until recently, access to light-sheet microscopes was largely restricted to research laboratories with the expertise required for building custom microscopes. However, with the market launch of various commercial light-sheet microscopes, such as the Carl Zeiss Lightsheet Z.1, this imaging technique is now available to essentially all researchers. As discussed above, our software modules can be applied to data sets produced with both custom and commercial microscopes.

As our laboratory consists of researchers with very diverse backgrounds, from mathematics and optical physics to biology, we took care to build our computational tools such that they can be used effectively without the need for a strong computational background. For example, image data in our KLB compression file format can be written and read through Fiji or MATLAB interfaces in exactly the same way that a TIFF file would be written or read. Our content-based MATLAB scripts for multiview image fusion are designed such that all configuration parameters are located in a simple MATLAB script that launches and manages each processing job automatically. Thus, the user essentially just needs to be familiar with the MATLAB interface itself and some basic commands for editing and running MATLAB scripts. When using computer clusters, a higher level of expertise is required in order to modify the respective support infrastructure provided by our software for submitting jobs in a given cluster environment.



Our segmentation and tracking software TGMM follows a similar design: the executable reads a configuration file that contains the parameters set by the user. Moreover, we provide executables that allow running our software out-of-the-box on Windows operating systems. Linux and Mac OS X users need to compile the code once to generate binaries, and thus some familiarity with CMake and C++ compilers is required for initial installation. All of these steps are documented in detail in our protocol and in the manuals included in our software packages.

The step that requires the most computational expertise is the setup of the CATMAID software: in addition to the installation of the application itself, the use of CATMAID requires setting up an HTTP server and a PostgreSQL database. We provide detailed documentation of these steps, but we also note that they are usually carried out by IT personnel or the system administrator of the academic institution. Once this initial setup is complete, users can simply interact with the program through a web browser, which does not require any particular expertise.

### Limitations

The segmentation and tracking modules of our processing pipeline were designed for cell tracking in images of nuclei-localized fluorescent markers. Shapes of cell nuclei in such images can



typically be well approximated as ellipsoid-like geometries<sup>39</sup>, and this assumption is reflected in the TGMM software by modeling the intensity profile of each nucleus as a 3D Gaussian. Thus, the TGMM software will typically not perform as well in images of objects with relatively irregular shapes, such as images of membrane markers. The other main requirement of the cell tracking protocol is that input image data should be well sampled along the time axis. As a rule of thumb, if an object moves between two consecutive time points by a distance larger than its diameter, the propagation of the associated 3D Gaussian shape parameters will probably not be successful. Finally, with regard to hardware limitations, execution of the TGMM framework requires a computer equipped with a CUDA-enabled nVidia graphics card.

## MATERIALS

### EQUIPMENT

#### Data files

- Data set 1, comprising example data for image masking and KLB image compression. This archive is available for download from our laboratory website (<https://www.janelia.org/lab/keller-lab/software>), and it contains a preconfigured version of the first module (clusterPT.m) of our MATLAB-based image processing pipeline for light-sheet microscopy data sets, all related auxiliary functions, a README file with software documentation and the folder Image\_Data with example data. The example data consist of a SiMView four-view recording (four image stacks with 125 images each) of a *Drosophila* embryo at an early developmental time point. The data set serve the purpose of illustrating image background masking and KLB lossless image compression with the MATLAB script clusterPT.m and follow the naming convention outlined in the README file. Note that clusterPT.m functionality also includes a dead pixel detector for removing respective image artifacts in scientific-grade complementary metal-oxide semiconductor (sCMOS) camera image data; however, dead pixels have already been corrected in this example data set.
- Data set 2, comprising example data for multiview image registration and fusion. This archive is available for download from our laboratory website (<https://www.janelia.org/lab/keller-lab/software>), and it contains preconfigured versions of the multiview image registration and fusion modules (clusterMF.m, localAP.m, clusterTF.m) of our MATLAB-based image processing pipeline for light-sheet microscopy data sets, all related auxiliary functions, a README file with software documentation and the folder Image\_Data with example data. The KLB-compressed example data consist of 11 time points of a SiMview four-view recording of an early *Drosophila* embryo processed with clusterPT.m. The data set serves the purpose of illustrating multiview image fusion of time-lapse light-sheet microscopy data with the MATLAB scripts clusterME.m, localAP.m and clusterTF.m.

#### Computer equipment

- Hardware requirements. For most benchmarks, the computational pipeline was deployed on a computer workstation equipped with two Intel Xeon E5-2687W CPUs, 192 GB DDR3 memory, an nVidia Tesla Kepler K20 GPU, six Seagate Savvio 10K.5 ST9900805SS hard disks combined in a RAID-6 data array, an Intel RMS25CB080 RAID module, an Intel X520-SR1 10Gb fiber network adapter and Windows 7 Professional 64 bit. For optimal processing speed, a good GPU and sufficient memory are of primary importance. The Tesla graphics card can be replaced with a lower-cost GeForce GTX Titan graphics card with little performance impact. Minimum requirements are an nVidia GPU with CUDA compute capability of 2.0 or higher. Information on CUDA compute capabilities of various GPUs is available at <https://developer.nvidia.com/cuda-gpus>. For a particularly cost-efficient build, slower CPUs and hard disks will generally suffice, as these components will only have a minor impact on processing speed.
- The performance benchmarks of the data compaction and multiview image fusion modules shown in **Table 1** were performed on a computer workstation equipped with two Intel Xeon E5-2667V2 CPUs, 256 GB DDR3 memory, an nVidia Quadro K2000D GPU, six Samsung 840 EVO 1 TB solid-state drives (SSDs) combined in a RAID-6 data array, an LSI 2208 RAID module, an Intel X520-SR1 10Gb fiber network adapter and Windows 8 Professional 64 bit.

As mentioned in earlier sections, there are a few additional limitations with respect to the other parts of our computational pipeline. First, our content-based multiview image fusion module does not support arbitrary optical geometries (please see ‘Applications of the protocol’ for details). Second, although the KLB lossless compression file format accepts a range of numerical data types (unsigned/signed integers, as well as floating point), best compression rates are typically obtained only for integer data types. With regard to hardware limitations, a computer with multicore CPU is required to take full advantage of the read and write speed improvements enabled by the block-based design of our file format. Finally, data visualization in CATMAID is limited to orthogonal cuts along the three axes of the underlying Cartesian coordinate system; i.e., the GUI does not render oblique slices of the image data.

- For data visualization, editing and annotation using CATMAID, a server with the following hardware components was used: two Intel Xeon E5-2690 CPUs, 128 GB of DDR3 memory, six Intel 520 Series 480 GB SSDs combined in a RAID-6 data array, an Intel RMS25CB080 RAID module, an Intel X520-SR1 10Gb fiber network adapter and the Linux distribution Ubuntu 12.04 LTS. Also in this case, slower CPUs and storage hardware will generally only have a minor performance impact. The SSDs constitute the most important hardware components as they ensure fast tile retrieval. We note that the same workstation can be used for CATMAID and for the rest of the computational pipeline.
- Software requirements. For several parts of our computational framework, a MATLAB installation (R2013b or later; MathWorks) is required, including the following toolboxes: Curve Fitting, Image Processing, Statistics, Optimization, Signal Processing and Parallel Computing. We verified compatibility specifically for MATLAB version R2013b, but our code should, in principle, be compatible with any version above R2011a, without a need for code modifications. We also note that the list of MATLAB toolbox requirements is based on the full functionality provided by our processing pipeline. Only a subset of these toolboxes is required to run the pipeline using typical parameter settings. A detailed overview of software and hardware requirements for all software packages is provided in **Supplementary Table 1**. Custom software packages are provided as **Supplementary Software 1–6**, and they can also be downloaded at <http://www.janelia.org/lab/keller-lab>.

#### EQUIPMENT SETUP

**Installation of TGMM software** Install the nVidia CUDA drivers included in the nVidia CUDA Toolkit available from <https://developer.nvidia.com/cuda-toolkit-archive>. If you are using a Linux Ubuntu distribution, simply execute the following terminal command:

```
sudo apt-get install nvidia-cuda-toolkit
```

To run the TGMM software (**Supplementary Software 4**), an nVidia graphics card with CUDA compute capability of 2.0 or higher is needed. Information about CUDA compute capability of all nVidia graphics cards is available at <https://developer.nvidia.com/cuda-gpus>.

#### Installing CATMAID for data visualization and cell-lineage editing

Download the latest version of the CATMAID branch for cell lineaging at [https://github.com/catmaid/CATMAID/tree/5d\\_cell\\_tracking](https://github.com/catmaid/CATMAID/tree/5d_cell_tracking) or clone it with the following Git command:

```
git clone -b 5Dvisualization --single-branch
https://fernandoamat@bitbucket.org/fernandoamat/
catmaid_5d_visualization_annotation.git
```

All installation details for Linux can be found in the user guide included in **Supplementary Software 5**, but we note that other operating systems can be used as well. Four main modules need to be set up: Django backend for running the web application CATMAID; HTTP server for web browsers for interacting with the backend; PostgreSQL database for storing all tracking information (but not for image data); and Image storage server for storing all image tiles.

It is possible to use a separate computer for storing image data and the database containing tracking information, as long as CATMAID has access to these data. CATMAID only needs to be installed once, and it can



subsequently be used through a web browser at any time from any location in the world with Internet access<sup>37</sup>. This step of the installation protocol requires the highest computational proficiency, and it is usually carried out by a system administrator or other IT personnel. In total, it should take ~1–3 h to configure all required software components.

**Optimizing HTTP server and PostgreSQL database configuration for optimal performance of the CATMAID web application** It is important to optimize the performance of the server in order to ensure the fastest possible interaction with CATMAID when visualizing image data and editing cell lineages through the web browser. Although there are many possible ways to optimize the system, we recommend in particular the following strategies that helped increase the performance of our system significantly:

We recommend using SSDs to store the image tiles. These drives should be mounted with the options ‘noatime’ and ‘nodiratime’ to avoid unnecessary read/write operations while serving image tiles to the web browser. Recommendations for further optimization can be found at <https://wiki.debian.org/SSDOptimization>.

If you are using the Linux partition format Ext2/Ext3, the i-node index descriptor is the main data structure describing files. Each node is

associated with one file and the block of addresses reserved for a file are stored in its index descriptor. However, the maximum number of i-nodes is set at the time of disk formatting and cannot be changed thereafter. Thus, if there are many small files, one can run out of i-nodes without running out of disk space. This scenario is possible for the image server because of the large number of tiles needed to partition large-scale data sets. Thus, we recommend accounting for an average file size of 4–8 kB when formatting the data partition of the image server. For example, in our system, the data array with a capacity of 1.7 TB was formatted using 268,435,456 i-nodes.

If the server has a large amount of RAM, the extent of data caching by the database and the operating system can be increased. Thereby, when users request the same image tiles multiple times, the server can retrieve them from memory instead of having to access the disk. To enhance caching, the following parameters need to be modified: ‘kernel.shmmax’ and ‘kernel.shmall’ in the file ‘/etc/sysctl.conf’ and ‘effective\_cache\_size’ and ‘shared\_buffers’ in the file ‘/etc/postgresql/X.X/main/postgresql.conf’. Recommendations for further optimization can be found at [http://wiki.postgresql.org/wiki/Tuning\\_Your\\_PostgreSQL\\_Server](http://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server).

## PROCEDURE

### Independent pipeline modules

**1** | The options described here focus on five classes of computational modules. Each of these modules can be executed independently or as part of a larger pipeline (**Figs. 1 and 2**):

Option	Module	Description
1A	Lossless image compression and/or multiview image fusion	We explain how large amounts of image data are efficiently stored and how arbitrary regions of interest in large image data are rapidly retrieved using our block-based lossless compression file format (KLB). We furthermore present MATLAB scripts for content-based registration and fusion of time-lapse, multiview image data
1B	Drift correction and intensity normalization	We discuss the use of our MATLAB scripts for drift correction and intensity normalization of time-lapse 3D stacks
1C	Segmentation and tracking with TGMM	We provide protocols for our TGMM software for automated large-scale segmentation and tracking of fluorescently labeled cell nuclei
1D	Data visualization and editing with CATMAID	We present a branch of CATMAID <sup>37,38</sup> that facilitates the visualization of five-dimensional microscopy data sets and allows editing associated cell tracking results
1E	Preparing videos for visualizing image data and cell lineage reconstructions	We describe MATLAB scripts for importing, analyzing and visualizing large-scale cell lineage reconstructions

### (A) Lossless compression of light-sheet microscopy data and/or multiview image fusion ● **TIMING** 5 min for setup, 0.5–12 h of unattended computer time (depending on data set size)

- Extract the test data and MATLAB scripts provided in Data set 1 (see ‘Data files’ in the MATERIALS section) to create a preconfigured test environment for performing background masking and/or lossless image compression using the KLB image format. The test data set included in this archive is a four-view image data set of a *Drosophila* embryo, which was recorded with a SiMView microscope.
- Open a MATLAB terminal and go to the folder containing the MATLAB scripts.
- Run the preconfigured MATLAB script clusterPT.m to verify proper software execution, and confirm that KLB output stacks are written to disk (output folder Image\_Data.corrected). Note that clusterPT.m can optionally also be configured to save output image data in an uncompressed TIFF file format (parameter ‘outputType’).

▲ **CRITICAL STEP** To run clusterPT.m on a new data set, use the code provided in **Supplementary Software 3** (comprising the complete MATLAB processing pipeline) and consult the software documentation (README file included with pipeline; see also parameter explanations provided in source code) to configure clusterPT.m for your data set (See **Box 1** for more information).

### ? TROUBLESHOOTING

■ **PAUSE POINT** At this point, the compressed image data can be manually inspected or imported into external software (proceed to Step 1A(iv)). If the data set is a multiview data set consisting of up to four views following the SiMView convention, image registration and fusion can now be performed by continuing with Step 1A(v). Spatial drift correction, intensity normalization or image filtering can be performed by continuing with Step 1B(i). If multiview

image fusion, as well as drift correction, normalization and/or filtering, is required, please follow the instructions for multiview image fusion first.

- (iv) Inspect the output image data generated by clusterPT.m. Once the images are stored in KLB format, they can be retrieved using the KLB C++ API provided in **Supplementary Software 1**. We also provide wrappers for MATLAB and Java, integration with Fiji<sup>64</sup> and an interface file for SWIG to autogenerate bindings for other languages (**Supplementary Software 1** and **2**). The KLB API provides efficient access to arbitrary regions of interest in the image volume by using block partitioning of the image data (**Supplementary Note**).
- (v) Extract the test data and MATLAB scripts provided in Data set 2 (see 'Data files' in the MATERIALS section) to create a preconfigured test environment for multiview image registration and fusion of SiMView-like image data sets with up to four views. The test data set included in this archive is a four-view image data set of a *Drosophila* embryo that was processed with clusterPT.m and stored in the KLB format.
- (vi) Execute the software modules for multiview image fusion. Multiview image registration and fusion consists of three steps (MATLAB scripts clusterMF.m, localAP.m and clusterTF.m) when processing time-lapse data sets. When processing individual image stacks rather than time-lapse data sets, only the first step (MATLAB script clusterMF.m) is required. In order to verify proper software execution and to get familiar with the full software functionality, run the preconfigured MATLAB scripts clusterMF.m, localAP.m and clusterTF.m included with the test data in sequential order. First, open a MATLAB terminal and go to the folder containing the scripts.  
**▲ CRITICAL STEP** To run these MATLAB scripts on new data sets, certain parameters will need to be adjusted (See **Box 2** for more information).
- (vii) Execute the first script, clusterMF.m. This script generates registered and fused image stacks, which are stored in the output folder 'Image\_Data.MultiFused'. The solution provided by clusterMF.m is not guaranteed to be smooth in time, as the data at each time point will be processed independently from the rest of the time-lapse data set.
- (viii) Execute the second script, localAP.m, to evaluate the registration results generated by clusterMF.m. This script produces smooth, interpolated parameter sets defining multiview image registration and multichannel/camera intensity matching transformations for all time points.
- (ix) Execute the third script, clusterTF.m. This script uses the information extracted by localAP.m and clusterMF.m in the previous two steps to perform temporally smooth multiview image fusion for the entire time-lapse data set.  
**▲ CRITICAL STEP** In the example data set, clusterMF.m is executed for all data points of the time-lapse experiment for demonstration purposes. When processing a large-scale time-lapse data set consisting of hundreds to thousands of time points recorded at high temporal resolution, we recommend running clusterMF.m only for a subset of time points (under typical conditions every tenth time point is sufficient) to save computation time and disk space. This sparse sampling of the time lapse data set is usually sufficient, as localAP.m will subsequently analyze and interpolate the results for smooth fusion of the entire time-lapse data set via clusterTF.m. The only exceptions to this rule are data sets in which temporal sampling is coarse and specimen shape and/or position changes drastically from one time point to the next. In this latter scenario, execution of clusterMF.m for all time points may improve image quality. It is important to keep this division of labor in mind, as execution of clusterMF.m is considerably more time-consuming per time point than execution of clusterTF.m.

## ? TROUBLESHOOTING

**■ PAUSE POINT** At this point, the fused image data can be manually inspected, used for data analysis or imported into external software. Spatial drift correction, intensity normalization or image filtering can be performed by continuing with Step 1B(i).

## Box 1 | Lossless compression and background masking of new image data sets

Verify that your input data follow the input data formatting requirements detailed in the README file and that all formatting parameters are correctly defined in clusterPT.m. KLB image compression is enabled or disabled via the parameter 'outputType'. Background masking is configured via the parameters 'segmentFlag' and 'thresholds'. The compressed image data, as well as associated foreground information and metadata, are stored in a new output folder, whose name is constructed by concatenating the input folder name with the extension '.corrected'. In addition, maximum-intensity projections of the output image data are stored in an output folder with the extension '.corrected.projections'.

Background masking is optional but important for maximum lossless data compression and data access speeds in subsequent steps of the pipeline (**Figs. 3** and **4**). The parameter 'thresholds' is crucial for achieving good foreground and background segmentation, and it should be carefully adjusted for each set of experiments to obtain optimal results. Note that background masking will overwrite background regions of the image data with zeros, but it will not alter image foreground (**Fig. 2**), unless the adaptive threshold level defined in 'thresholds' is set too high.

## Box 2 | Multiview fusion of new image data sets

To run the scripts on a new data set, use the code provided in **Supplementary Software 3** (comprising the complete MATLAB processing pipeline) and consult the software documentation (README file included with the pipeline; see also parameter explanations provided in source code) to configure each script for your data set. Verify that all formatting parameters are correctly defined.

A few critical parameters may need to be changed in each script. In particular, formatting parameters defining the location and properties of the input image data generally need to be updated for each new run. We also note that, if background masking was disabled in the preceding clusterPT.m processing step, the parameter 'maskFactor' is used to define the adaptive threshold level for background masking in clusterMF.m. Background masking in one of the two modules is required for estimating the geometrical specimen outline. This information is needed for modeling the illumination and detection path lengths inside the sample and, thus, for rapid and optimal assessment of relative image quality in the various views as a function of location in the specimen. clusterMF.m and clusterTF.m furthermore provide the parameter 'fusionType' for defining the type of image fusion applied after image registration (options include adaptive blending, geometrical blending, wavelet fusion and averaging). Adaptive and geometrical blending are stitching methods, whereas averaging and wavelet fusion use information from the entire volume in all views. We generally recommend blending for optimal image quality and processing speed in SiMView-type four-view data sets. Wavelet fusion is computationally much more costly, and it has a tendency of introducing fusion artifacts, such as ringing, but it maintains the same signal-to-noise ratio throughout the image volume, including the region in which stitching methods would otherwise introduce a blending seam. Averaging is comparable to stitching with respect to processing speed, and it can be more robust when specimens are only very sparsely labeled, but image quality is generally inferior to stitching methods. Information about the microscope setup, specifically the relative orientation of light sheets and cameras, is provided in the parameters 'leftFlags', 'flipHFlag', 'flipVFlag', 'frontFlag', 'xOffsets' and 'yOffsets'. If the data set is not a time-lapse data set, the scripts localAP.m and clusterTF.m can be skipped.

### (B) Spatial drift correction, intensity normalization and image filtering ● **TIMING** 10 min for setup, 0.5–12 h of unattended computer time (depending on data set size)

- (i) Extract the code provided in **Supplementary Software 3** (comprising the complete MATLAB processing pipeline) in order to start using the software modules for 3D spatial drift correction, intensity normalization throughout a time-lapse image data set and/or image filtering for adaptive local background correction. The output from Step 1A(ix) can be used in this section as an example. Proceed to Step 1B(ii) for drift correction and/or intensity normalization. Proceed to Step 1B(iv) for image filtering for adaptive local background correction.
- (ii) Consult the software documentation (README file included with pipeline; see also parameter explanations provided in source code) to configure and run localEC.m. localEC.m is a data analysis script that preprocesses the time-lapse data set for subsequent 3D spatial drift correction and/or intensity normalization with clusterCS.m in Step 1B(iii). Verify that all formatting parameters are correctly defined. localEC.m provides the parameters 'intensityFlag' to enable/disable intensity normalization and 'correlationFlag' to enable/disable 3D drift correction.
- (iii) Run clusterCS.m script. Once the corresponding intensity/drift information has been collected by localEC.m (previous step), compensatory image adjustments can subsequently be applied by clusterCS.m using the parameters 'correctDrift' to execute drift correction (using the parameter 'referenceTime' as a temporal anchor, that is, as the time point relative to which adjustments of data at all other time points are performed) and 'correctIntensity' to execute intensity normalization.
 

▲ **CRITICAL STEP** The scripts localEC.m and clusterCS.m use multiple complementary strategies to estimate short-term specimen fluctuations and long-term specimen drift, respectively. The former is computed via image correlation (which provides accurate frame-to-frame corrections but can introduce long-term drift), whereas the latter is estimated based on computation of the geometrical center of the specimen (which captures long-term drift but is too noisy for frame-to-frame corrections). The combination of both methods provides optimal short-term and long-term drift correction, and it is enabled by setting the parameter 'globalMode' to 1.
- (iv) Consult the software documentation (README file included with pipeline; see also parameter explanations provided in source code) to configure and run clusterFR.m for your data set. clusterFR.m uses Gaussian filtering for adaptive local background subtraction and generates filtered image stacks and/or maximum-intensity projections of filtered image stacks. The radius used for anisotropic Gaussian filtering is defined in the parameter 'rangeArray'.
 

▲ **CRITICAL STEP** Note that clusterFR.m is implemented primarily for image visualization purposes and, owing to the local nature of the image corrections, it is not recommended in a workflow for quantitative image analysis.

■ **PAUSE POINT** At this point, the drift-corrected, normalized and/or filtered image data can be manually inspected, used for further data analysis or imported into external software.



## **(C) Automated segmentation and tracking with TGMM** ● **TIMING** 5 min for setup, 0.5–5 h of unattended computer time (depending on data set size)

▲ **CRITICAL STEP** The protocol described here describes how to run ‘TGMM’ for the test data set included in **Supplementary Software 4** to verify that it executes correctly on your workstation. Users of Windows 7 64-bit machines can directly use the precompiled binaries located in the folder ‘bin’. Users of other operating systems, such as Linux, first need to compile the code according to the instructions provided in the README file. In order to run the software on a new data set, use the configuration file ‘TGMM\_configFile.txt’ provided with the test data set as a template and modify parameters as needed (See **Box 3** for details).

- (i) Run the program ‘ProcessStack’ to generate a hierarchical segmentation for each time point. The software documentation explains how to parallelize the execution of this program on all time points using simple scripts in Unix and Windows.

▲ **CRITICAL STEP** Note that running the hierarchical segmentation algorithm in parallel for multiple time points might use all available computing resources. Thus, while the segmentation algorithm is running, the performance of other applications on this computer may be affected. The TGMM software package also includes the program ‘ProcessStack\_woGPU’, which offers the same functionality as ‘ProcessStack’ but does not require an nVidia GPU. This executable is useful for distributing the hierarchical segmentation task in cluster environments.

▲ **CRITICAL STEP** Step 1C(i) only needs to be repeated if the parameter ‘backgroundThreshold’ or any of the advanced parameters in the hierarchical segmentation category are changed in the file ‘TGMM\_configFile.txt’. Otherwise, the existing binary files can be reused to run the tracking module multiple times with different parameter settings.

■ **PAUSE POINT** Segmentation results are stored in binary files with the suffix ‘hierarchicalSegmentation’ (one per processed time point) in the same folder as the original image. These binary files contain all possible segmentations for different values of ‘persistenceSegmentationTau’. Proceed to Step 1C(ii) to continue with automated cell tracking.

- (ii) Run the program ‘TGMM’ to segment and track cells for all time points. This algorithm uses the binary files generated in the previous step to define super-voxels.

### **? TROUBLESHOOTING**

■ **PAUSE POINT** At this point, cell lineaging results are stored as XML files (one per processed time point). Proceed to Step 1D(i) to continue with data visualization and editing of the automatically generated tracking results, or proceed to Step 1E(i) to continue with the analysis of cell tracks.

## **(D) Visualizing and editing lineaging results using CATMAID** ● **TIMING** 0.5–4 h (depending on data set size)

▲ **CRITICAL STEP** The protocol described here explains how to use the CATMAID browser interface for visualizing and editing cell tracks. Before executing this protocol, make sure to install and configure the CATMAID backend service according to the instructions in Equipment Setup.

### **Box 3 | TGMM parameter optimization**

The threshold for persistence-based agglomeration of watershed regions (‘persistenceSegmentationTau’) and the intensity threshold for defining the background level of the image data (‘backgroundThreshold’) are the two most important adjustable parameters of the TGMM software. Both relate to image properties and are straightforward to determine by visual inspection of the image data at a late time point of the time-lapse recording. We found that inspecting late time points is generally most useful, in particular if intensity levels become lower and cell densities become higher as time progresses. Measurements in this latter scenario provide a lower bound constraint for both parameters.

To determine the background threshold, inspect a region of the image volume located outside of the specimen (for example, by using the open-source software ImageJ<sup>65</sup>) and measure the mean intensity level in this background region. It is preferable to be conservative in this assessment—i.e., to set the background threshold to a relatively low level. This setting minimizes the number of missed cell nuclei and reduces false negative detections, which can otherwise affect coherence between time points. To determine the threshold for persistence-based agglomeration of watershed regions, plot the intensity profile along a line connecting the centroids of two of the dimmest nuclei in the image stack (for example, by using ImageJ<sup>65</sup>). The profile should exhibit two peaks (nuclei centroids) and a valley in between (nuclei boundaries). The threshold should be set to a value smaller than the difference between the intensity values of the peaks and the valley. This setting ensures that the corresponding nuclei are not merged to a single super-voxel (undersegmentation). In our experience, a value between 5 and 20 of the parameter ‘persistenceSegmentationTau’ tends to be sufficient to compensate for watershed oversegmentation of noisy regions, without risking merging of dim cell nuclei. We furthermore note that one can usually obtain close-to-optimal results for a fairly wide range of parameter values<sup>39</sup>, although care should be taken to set these parameters appropriately.

- (i) Run the MATLAB script 'generateTilesFromFolder' provided in **Supplementary Software 6** to transform all image stacks from Step 1A(ix) into sets of tiles that can be read and requested by the browser through its connection to CATMAID. The README file accompanying the script, as well as the user guide, provides instructions for setting each parameter.

▲ **CRITICAL STEP** The script needs to be executed on a computer with write access to the image server in order to save the newly generated image tiles.

? **TROUBLESHOOTING**

- (ii) Log on to the CATMAID administrator website and select the option 'Add' in the Stack menu. A form requesting details about the image data generated in Step 1D(i) will appear in the browser. Completing this form creates a new entry in the CATMAID database with information about location and attributes of the image tiles. The documentation at [http://catmaid.org/importing\\_data.html](http://catmaid.org/importing_data.html) provides more details on how to perform this step.

▲ **CRITICAL STEP** The parameter 'Tile source type' needs to be set to 5 to inform CATMAID that the images contain temporal information.

▲ **CRITICAL STEP** The parameters 'Num zoom levels' and 'Tile size' need to match the settings specified in the previous step.

■ **PAUSE POINT** Partitioning image data into tiles only needs to be done once for each data set. Tiles are stored in the image server accessible by CATMAID and reused every time a new set of cell tracks is uploaded, thus avoiding image data duplication.

- (iii) Log on to the CATMAID administrator website and select the option 'TGMM importer' in the Custom Views menu.
- (iv) Specify where the XML files are located (field 'Xml basename') and which image data set should be associated with the XML files ('Dataset id' from Step 1D(ii)). The field 'Project name' allows assigning a unique name to this cell-lineage reconstruction.

▲ **CRITICAL STEP** The XML output files from Step 1C(ii) need to be copied to a location at which CATMAID can read from.

? **TROUBLESHOOTING**

■ **PAUSE POINT** Tracking information is stored in the CATMAID database, and it can be edited, analyzed or visualized at any time.

- (v) Open a browser and enter the URL of the web application CATMAID containing your data.
- (vi) Select the project that you would like to work on.

? **TROUBLESHOOTING**

- (vii) Use the sliders on the toolbar to navigate the image data in five dimensions. In order to visualize and manipulate the cell tracking information, select the 'Tracing tool' in the toolbar. Note that a click on the '?' icon will display all possible actions in each view. CATMAID offers many different types of editing and visualization operations for the cell tracking data (add/delete edge or point, display lineage, show orthogonal planes, etc.). A comprehensive documentation of all functionality can be found at <http://catmaid.org/> and in the user guide included in **Supplementary Software 5**.

**Supplementary Software 5.**

▲ **CRITICAL STEP** We recommend periodic backups of the CATMAID database (at least once a week) in order to minimize the risk of data loss. As the database only contains points in object space (i.e., no image data), the size of these backups is typically fairly small.

■ **PAUSE POINT** This step can be interrupted at any time. Every time an operation is performed by the user, the change is immediately stored in the CATMAID database and an entry is added to the log table. Thus, work on a specific project can be resumed at any time.

**(E) Importing tracking information into MATLAB and preparing videos for visualizing image data and cell-lineage reconstructions ● TIMING 5 min**

▲ **CRITICAL STEP** Segmentation and tracking results can be imported into MATLAB at different stages of the pipeline. If you would like to analyze the output from Step 1D(vii), then proceed to Step 1E(ii). If you would like to analyze the output from Step 1C(ii), then proceed to Step 1E(iii).

- (i) Extract the contents of the archive provided in **Supplementary Software 6** or download the MATLAB scripts for interacting with the PostgreSQL database accessed by CATMAID for storing lineage information using the following Git command:

```
git clone https://fernandoamat@bitbucket.org/fernandoamat/catmaid-matlab-code.git
```

- (ii) Use the MATLAB script 'scriptRetrieveNodeWithTag' to import all data points of a specific project from the CATMAID database into MATLAB. The README file accompanying the script provides documentation on how to set each of the

required parameters. The script returns an  $N \times 10$  MATLAB array, where  $N$  is the number of data points in the database for the requested project. The ten columns contain the following information:

Column	Contents
1	Unique ID used to identify the cell nucleus data entry in the database (a large integer number)
2	Cell type (represented by an integer). The value is set to 0 if no cell type has been assigned to this object
3	$x$ location of the nucleus centroid in world coordinates. Use the resolution information provided in variable 'stackRes' to convert from world coordinates to pixels
4	Same as 3 but for $y$ location
5	Same as 3 but for $z$ location
6	Estimated radius of the nucleus. The value is set to 0 if a radius estimate is not available
7	ID of the corresponding cell at the previous time point. The value is set to $-1$ if there is no linkage to the previous time point. Otherwise, the value is set to the unique ID of the parent of the cell identified in column 1. Following the path of these parent/daughter assignments allows reconstructing the complete cell lineage information
8	Experiment time point at which the cell nucleus has been detected
9	Confidence level for the tracking result. A value of 3 indicates high confidence that the object was correctly tracked. A value of 0 indicates low confidence
10	Skeleton ID. All data points belonging to the same cell lineage have the same unique skeleton ID

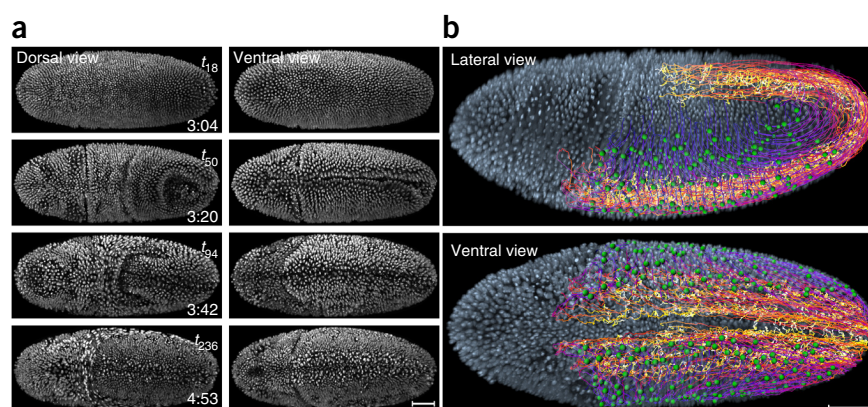
- (iii) Import tracking information from TGMM into MATLAB. The output from Step 1C(v) is stored in XML files that can be imported into the same type of MATLAB array, as described in the previous step. This import functionality is provided by the script 'parseMixtureGaussiansXml2trackingMatrixCATMAIDformat'. We note, however, that this MATLAB array format does not cover all of the information present in the XML files. For example, super-voxel segmentation information is lost. To import all available information into MATLAB, execute the script 'readXMLmixtureGaussians' for each time point. This script returns a structure mimicking the attributes of the XML file for each database object. More information about the various attributes can be found in the documentation accompanying the script.

**▲ CRITICAL STEP** The XML files were generated using C++ code, in which indices start at 0. In contrast, MATLAB convention has indices start at 1. Thus, for code written in MATLAB, all indices need to be offset by +1. For example, if the parent ID is 3, the index 4 needs to be used in MATLAB in order to retrieve the correct cell-lineage information from the MATLAB structure.

- (iv) Prepare videos for visualizing image data and cell-lineage reconstructions. Once the data have been imported into MATLAB as an  $N \times 10$  array (as described in Step 1E(iii)), the cell-lineage information can be visualized in a variety of ways. For data visualization using the commercial software package Imaris, we provide export scripts that enable

**Figure 7 | Application example in *Drosophila* development.** (a) Maximum-intensity projections of image data at four time points of a SiMView time-lapse data set of *Drosophila* embryonic development<sup>8,39</sup> (left, dorsal view; right, ventral view). The complete data set comprises 2,881 time points (each consisting of a four-view recording of the embryo with  $4 \times 154$  images) recorded in 30-s intervals, and it was processed with the pipeline presented in **Figures 1 and 2**. The total data set size is 4.35 TB (1,774,696 images). The age of the nuclei-labeled (His2Av-mRFP1) embryo is shown in hours and minutes after egg laying in the bottom right corner of the dorsal view panels.

(b) Visualization of a cell-lineage reconstruction of early nervous system development in the *Drosophila* ventral nerve cord. The cell-lineage reconstruction was performed with TGMM and CATMAID, using the data set visualized in **a**. The data were rendered with Imaris, using microscopy image data at time point ( $t$ ) 50 (see second row in **a**). Green spheres represent the positions of progenitor cell nuclei at time point 50. Lines represent complete cell tracks using a color code to indicate time (from purple to yellow: 2.9–5.4 h after egg laying). Scale bars, 50  $\mu$ m.





the data transfer of cell tracks to Imaris. This approach enables joint rendering of microscopy data and cell-lineage information (**Fig. 7**). The ImarisXT interface and the scripts provided in **Supplementary Software 6** are required to establish communication between MATLAB and Imaris.

- (v) Open a single Imaris session (if there is more than one active Imaris session, the MATLAB script will not know which session the data need to be exported to).
- (vi) In Imaris, open the time-lapse microscopy image data used for cell tracking, and click on the Surpass Scene folder so that it is highlighted.
- (vii) Without closing Imaris, switch to a MATLAB terminal and import the cell tracking data, as described in Step 1E(iii).
- (viii) Execute the MATLAB script 'parseCATMAIDdbToImarisMultiSpots'. Once the script execution has finished, a new Imaris spot object will appear in the Imaris window for each imported cell lineage.

### ? TROUBLESHOOTING

- (ix) Save the Imaris project to store the imported cell lineage information.

### ? TROUBLESHOOTING

Troubleshooting advice can be found in **Table 3**.

**TABLE 3** | Troubleshooting table.

Step	Problem	Possible reason(s)	Solution
1A(iii)	MATLAB script clusterPT.m stops with an error displayed in MATLAB command window	Input data are not correctly formatted for clusterPT.m	Make sure that your image data set has been formatted according to the specifications provided in the README file (see section 'Part I') and that background images and XML files are present as well
1A(ix)	The MATLAB scripts clusterMF.m, localAP.m or clusterTF.m stop with an error displayed in the MATLAB command window	Data paths or image channel/camera configuration are set incorrectly	Check the parameters defining the data locations ('inputString', 'outputString', 'sourceString', 'inputID', 'outputID', 'lookUpTable') and image channel/camera configuration ('specimen', 'cameras', 'channels', 'sChannels', 'tChannels') in clusterMF.m, localAP.m and clusterTF.m
	Image fusion of a time-lapse multiview data set produces images that are not smooth in time	Smoothing window size in localAP.m is too small. Time points are sampled too infrequently in clusterMF.m	Increase the smoothing window size in localAP.m (second value in vector smoothing) and rerun clusterTF.m. The effects of changing the smoothing window size or the averaging ranges defined in localAP.m are shown in the MATLAB plots displayed for each transformation parameter (blue lines indicate input data collected by clusterMF.m, red dashed lines indicate smooth output data provided to clusterTF.m). If adjusting smoothing and averaging parameters in localAP.m does not resolve the issue, try increasing the sampling frequency of the time-lapse experiment in clusterMF.m and rerun localAP.m and clusterTF.m
1C(ii)	TGMM software stops with a run-time error displayed in the terminal window	nVidia CUDA drivers are not up-to-date. Image path is incorrect	Consult the section 'Troubleshooting common run-time errors' in the README file included in <b>Supplementary Software 4</b> . This section provides solutions to the most commonly encountered TGMM run-time errors
1D(i)	The MATLAB script for image tiling stops with an error displayed in the MATLAB command window	The user does not have write permission to the image server associated with CATMAID	Make sure to enable write permissions (for the output path provided in the MATLAB script) for the user executing the tiling script

(continued)

PROTOCOL

TABLE 3 | Troubleshooting table (continued).

Step	Problem	Possible reason(s)	Solution
1D(iv)	CATMAID TGMM importer displays error message ‘504 Gateway Timeout Error’	The amount of time needed for importing a large number of cell tracks can exceed the default timeout setting of the browser	If the number of tracked objects is relatively large, it is usually not possible to upload all data to the CATMAID database before encountering a browser timeout. However, even if a timeout message is being displayed, the data are still being uploaded. The file ‘progressReportFile.txt’, located in the same folder as the XML output files, monitors the status of the data transfer. You can consult this log file to determine when all data points have been uploaded to the database
1D(vi)	The CATMAID home page does not show your project	The user is not logged in or does not have permission to access the project	Make sure that you are logged in as a user (this information is provided in the top right corner of the browser window) and that you have permission to access the project
1E(viii)	The MATLAB script for exporting tracks to Imaris stops with an error message	The path to Imaris XT Java library is set incorrectly in the MATLAB script	Open the MATLAB file ‘openImarisConnection.m’ and verify that line 5 contains the correct path to the Imaris XT Java library

● TIMING

Step 1A, lossless image compression and/or multiview image fusion: 5 min for setup, and 0.5–12 h of unattended computer time (depending on data set size)  
Step 1B, drift correction and intensity normalization: 10 min for setup, and 0.5–12 h of unattended computer time (depending on data set size)  
Step 1C, segmentation and tracking with TGMM: 5 min for setup, and 0.5–5 h of unattended computer time (depending on data set size)  
Step 1D, data visualization and editing with CATMAID: 0.5–4 h (depending on data set size)  
Step 1E, preparing videos for visualizing image data and cell-lineage reconstructions: 5 min

ANTICIPATED RESULTS

By following the steps in this protocol carefully, the user should expect to be able to convert microscopy data sets of animal development with up to several terabytes of image data per experiment into curated cell-lineage reconstructions (Fig. 7). A single computer workstation is sufficient for the routine use of the presented computational framework at all stages of the protocol. The raw microscopy image data are efficiently stored in the KLB lossless image compression format. If multiview image data recorded with a SiMView-like microscope is used as a starting point, our pipeline enables rapid content-based multiview registration and fusion. Together, these steps reduce typical multiterabyte light-sheet microscopy data sets to a few tens of gigabytes of compressed image data, thus efficiently eliminating data storage bottlenecks without data loss. After these initial data management steps, our image processing and data analysis protocol leads to a CATMAID database that contains comprehensive cell-lineage information for the reconstructed data set, including a log of all edits performed by each user. The MATLAB scripts included in this protocol can be used to visualize final results and to generate publication-quality videos. They also facilitate data import into a MATLAB array, which gives access to a wide spectrum of further analyses, such as computation of cell-lineage statistics or investigation of morphodynamic features, including cell cycle lengths, cell velocities and temporal changes in gene expression levels in each tracked cell lineage.

When using KLB lossless compression in combination with background masking, a 10–200-fold reduction in data size can be expected for typical fluorescence microscopy recordings (Figs. 3 and 4; and 30- to 500-fold with additional multiview image fusion). Aside from reducing storage capacity needs, KLB image compression also helps improve data transfer rates and access speed. The TGMM software is capable of segmenting and tracking ~26,000 cells per minute on a single workstation<sup>39</sup>, with an average linkage accuracy ranging from 99% for zebrafish embryos to 90% for mouse embryos.

In our experience, novice users can usually familiarize themselves with the overall pipeline within a few days (see ‘Level of expertise needed to implement the protocol’ in the INTRODUCTION for specific details), as most of the scripts used in this protocol simply require adjusting a few parameters. If completely error-free cell-lineage reconstructions are required,

the most time-intensive part of the protocol is the manual curation of cell -lineage results produced by the TGMM software. This task is facilitated by the web application CATMAID, which enables typical data curation rates of ~1,400 data points per hour and thus offers the possibility of system-level cell lineaging. For example, a fully curated cell-lineage reconstruction of the early *Drosophila* nervous system, tracking 92% of S1 neuroblasts from their blastoderm origins up to their second cell division (over 116,000 data points spanning more than 300 time points), was performed within 3 weeks<sup>39</sup>.

Note: Any Supplementary Information and Source Data files are available in the online version of the paper.

**ACKNOWLEDGMENTS** We thank A. Cardona and the participants of the Janelia CATMAID hackathon for help with modifying the open-source code of CATMAID; K. Khairy for his contributions to exploring approaches to multiview image fusion and SiMView data management; and K. Branson and A. Cardona for helpful comments on the manuscript. This work was supported by the Howard Hughes Medical Institute.

**AUTHOR CONTRIBUTIONS** F.A. and B.H. developed the KLB file format and related software infrastructure. P.J.K. developed the multiview registration and fusion software, with contributions from F.A. F.A. developed the TGMM framework and related software infrastructure. Y.W., W.C.L. and K.M. performed light-sheet microscopy experiments and contributed image data sets. F.A. and P.J.K. wrote the manuscript, with input from all authors.

**COMPETING FINANCIAL INTERESTS** The authors declare no competing financial interests.

Reprints and permissions information is available online at <http://www.nature.com/reprints/index.html>.

- Voie, A.H., Burns, D.H. & Spelman, F.A. Orthogonal-plane fluorescence optical sectioning: three-dimensional imaging of macroscopic biological specimens. *J. Microsc.* **170**, 229–236 (1993).
- Fuchs, E., Jaffe, J., Long, R. & Azam, F. Thin laser light sheet microscope for microbial oceanography. *Opt. Express* **10**, 145–154 (2002).
- Huisken, J., Swoger, J., Del Bene, F., Wittbrodt, J. & Stelzer, E.H.K. Optical sectioning deep inside live embryos by selective plane illumination microscopy. *Science* **305**, 1007–1009 (2004).
- Keller, P.J., Schmidt, A.D., Wittbrodt, J. & Stelzer, E.H. Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy. *Science* **322**, 1065–1069 (2008).
- Ahrens, M.B., Orger, M.B., Robson, D.N., Li, J.M. & Keller, P.J. Whole-brain functional imaging at cellular resolution using light-sheet microscopy. *Nat. Methods* **10**, 413–420 (2013).
- Wu, Y. *et al.* Spatially isotropic four-dimensional imaging with dual-view plane illumination microscopy. *Nat. Biotechnol.* **31**, 1032–1038 (2013).
- Krzic, U., Gunther, S., Saunders, T.E., Streichan, S.J. & Hufnagel, L. Multiview light-sheet microscope for rapid *in toto* imaging. *Nat. Methods* **9**, 730–733 (2012).
- Tomer, R., Khairy, K., Amat, F. & Keller, P.J. Quantitative high-speed imaging of entire developing embryos with simultaneous multiview light-sheet microscopy. *Nat. Methods* **9**, 755–763 (2012).
- Schmid, B. *et al.* High-speed panoramic light-sheet microscopy reveals global endodermal cell dynamics. *Nat. Commun.* **4**, 2207 (2013).
- Holekamp, T.F., Turaga, D. & Holy, T.E. Fast three-dimensional fluorescence imaging of activity in neural populations by objective-coupled planar illumination microscopy. *Neuron* **57**, 661–672 (2008).
- Truong, T.V., Supatto, W., Koos, D.S., Choi, J.M. & Fraser, S.E. Deep and fast live imaging with two-photon scanned light-sheet microscopy. *Nat. Methods* **8**, 757–760 (2011).
- Gao, L. *et al.* Noninvasive imaging beyond the diffraction limit of 3D dynamics in thickly fluorescent specimens. *Cell* **151**, 1370–1385 (2012).
- Chen, B.C. *et al.* Lattice light-sheet microscopy: imaging molecules to embryos at high spatiotemporal resolution. *Science* **346**, 1257998 (2014).
- Keller, P.J. *et al.* Fast, high-contrast imaging of animal development with scanned light sheet-based structured-illumination microscopy. *Nat. Methods* **7**, 637–642 (2010).
- Capoulade, J., Wachsmuth, M., Hufnagel, L. & Knop, M. Quantitative fluorescence imaging of protein diffusion and interaction in living cells. *Nat. Biotechnol.* **29**, 835–839 (2011).
- Keller, P.J. Imaging morphogenesis: technological advances and biological insights. *Science* **340**, 1234168 (2013).
- Pantazis, P. & Supatto, W. Advances in whole-embryo imaging: a quantitative transition is underway. *Nat. Rev. Mol. Cell Biol.* **15**, 327–339 (2014).
- Stelzer, E.H. Light-sheet fluorescence microscopy for quantitative biology. *Nat. Methods* **12**, 23–26 (2014).
- Huisken, J. Slicing embryos gently with laser light sheets. *Bioessays* **34**, 406–411 (2012).
- Pampaloni, F., Reynaud, E.G. & Stelzer, E.H. The third dimension bridges the gap between cell culture and live tissue. *Nat. Rev. Mol. Cell Biol.* **8**, 839–845 (2007).
- Keller, P.J., Ahrens, M.B. & Freeman, J. Light-sheet imaging for systems neuroscience. *Nat. Methods* **12**, 27–29 (2014).
- Keller, P.J. & Ahrens, M.B. Visualizing whole-brain activity and development at the single-cell level using light-sheet microscopy. *Neuron* **85**, 462–483 (2015).
- Lemon, W.C. & Keller, P.J. Live imaging of nervous system development and function using light-sheet microscopy. *Mol. Reprod. Dev.* **82**, 605–618 (2015).
- Megason, S.G. & Fraser, S.E. Imaging in systems biology. *Cell* **130**, 784–795 (2007).
- Khairy, K. & Keller, P.J. Reconstructing embryonic development. *Genesis* **49**, 488–513 (2011).
- McMahon, A., Supatto, W., Fraser, S.E. & Stathopoulos, A. Dynamic analyses of *Drosophila* gastrulation provide insights into collective cell migration. *Science* **322**, 1546–1550 (2008).
- Fernandez, R. *et al.* Imaging plant growth in 4D: robust tissue reconstruction and lineaging at cell resolution. *Nat. Methods* **7**, 547–553 (2010).
- Bosveld, F. *et al.* Mechanical control of morphogenesis by Fat/Dachsous/Four-jointed planar cell polarity pathway. *Science* **336**, 724–727 (2012).
- Murray, J.I. *et al.* Automated analysis of embryonic gene expression with cellular resolution in *C. elegans*. *Nat. Methods* **5**, 703–709 (2008).
- Liu, X. *et al.* Analysis of cell fate from single-cell gene expression profiles in *C. elegans*. *Cell* **139**, 623–633 (2009).
- Trichas, G. *et al.* Multi-cellular rosettes in the mouse visceral endoderm facilitate the ordered migration of anterior visceral endoderm cells. *PLoS Biol.* **10**, e1001256 (2012).
- Xiong, F. *et al.* Specified neural progenitors sort to form sharp domains around noisy Shh signaling. *Cell* **153**, 550–561 (2013).
- Du, Z., Santella, A., He, F., Tiongson, M. & Bao, Z. *De novo* inference of systems-level mechanistic models of development from live-imaging-based phenotype analysis. *Cell* **156**, 359–372 (2014).
- Panier, T. *et al.* Fast functional imaging of multiple brain regions in intact zebrafish larvae using selective plane illumination microscopy. *Front. Neural Circuits* **7**, 65 (2013).
- Lemon, W. *et al.* Whole central nervous system functional imaging in larval *Drosophila*. *Nat. Commun.* **6**, 7924 (2015).
- Alivisatos, A.P. *et al.* The brain activity map project and the challenge of functional connectomics. *Neuron* **74**, 970–974 (2012).
- Saalfeld, S., Cardona, A., Hartenstein, V. & Tomancák, P. CATMAID: collaborative annotation toolkit for massive amounts of image data. *Bioinformatics* **25**, 1984–1986 (2009).
- Cardona, A. Collaborative annotation toolkit for massive amounts of image data (CATMAID) GitHub repository <https://github.com/acardona/CATMAID> (2015).
- Amat, F. *et al.* Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data. *Nat. Methods* **11**, 951–958 (2014).
- Lauri, A. *et al.* Development of the annelid axochord: insights into notochord evolution. *Science* **345**, 1365–1368 (2014).
- Preibisch, S., Saalfeld, S., Schindelin, J. & Tomancák, P. Software for bead-based registration of selective plane illumination microscopy data. *Nat. Methods* **7**, 418–419 (2010).
- Bao, Z. *et al.* Automated cell lineage tracing in *Caenorhabditis elegans*. *Proc. Natl. Acad. Sci. USA* **103**, 2707–2712 (2006).



43. Murray, J.I., Bao, Z., Boyle, T.J. & Waterston, R.H. The lineaging of fluorescently-labeled *Caenorhabditis elegans* embryos with StarryNite and AceTree. *Nat. Protoc.* **1**, 1468–1476 (2006).
44. Giurumescu, C.A. *et al.* Quantitative semi-automated analysis of morphogenesis with single-cell resolution in complex embryos. *Development* **139**, 4271–4279 (2012).
45. Olivier, N. *et al.* Cell lineage reconstruction of early zebrafish embryos using label-free nonlinear microscopy. *Science* **329**, 967–971 (2010).
46. Kausler, B.X. *et al.* A discrete chain graph model for 3D+t cell tracking with high misdetection robustness. *ECCV* **7574**, 144–157 (2012).
47. Stegmaier, J. *et al.* Fast segmentation of stained nuclei in terabyte-scale, time resolved 3D microscopy image stacks. *PLoS ONE* **9**, e90036 (2014).
48. Schiegg, M. *et al.* Graphical model for joint segmentation and tracking of multiple dividing cells. *Bioinformatics* **31**, 948–956 (2014).
49. Allan, C. *et al.* OMERO: flexible, model-driven data management for experimental biology. *Nat. Methods* **9**, 245–253 (2012).
50. Megason, S.G. *In toto* imaging of embryogenesis with confocal time-lapse microscopy. *Methods Mol. Biol.* **546**, 317–332 (2009).
51. Schroeder, W., Martin, K. & Lorensen, B. *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. 4th edn. (Kitware, 2006).
52. Peng, H., Ruan, Z., Long, F., Simpson, J.H. & Myers, E.W. V3D enables real-time 3D visualization and quantitative analysis of large-scale biological image data sets. *Nat. Biotechnol.* **28**, 348–353 (2010).
53. Bria, A., Iannello, G. & Peng, H. An open-source VAA3D plugin for real-time 3D visualization of terabyte-sized volumetric images. *ISBI*, 520–523 (2015).
54. Pietzsch, T., Saalfeld, S., Preibisch, S. & Tomancak, P. BigDataViewer: visualization and processing for large image data sets. *Nat. Methods* **12**, 481–483 (2015).
55. Akerboom, J. *et al.* Optimization of a GCaMP calcium indicator for neural activity imaging. *J. Neurosci.* **32**, 13819–13840 (2012).
56. Chen, T.W. *et al.* Ultrasensitive fluorescent proteins for imaging neuronal activity. *Nature* **499**, 295–300 (2013).
57. Kanodia, J.S. *et al.* A computational statistics approach for estimating the spatial range of morphogen gradients. *Development* **138**, 4867–4874 (2011).
58. Pitrone, P.G. *et al.* OpenSPIM: an open-access light-sheet microscopy platform. *Nat. Methods* **10**, 598–599 (2013).
59. Gualda, E.J. *et al.* OpenSpinMicroscopy: an open-source integrated microscopy platform. *Nat. Methods* **10**, 599–600 (2013).
60. Bock, D.D. *et al.* Network anatomy and *in vivo* physiology of visual cortical neurons. *Nature* **471**, 177–182 (2011).
61. Tomer, R., Ye, L., Hsueh, B. & Deisseroth, K. Advanced CLARITY for rapid and high-resolution imaging of intact tissues. *Nat. Protoc.* **9**, 1682–1697 (2014).
62. Susaki, E.A. *et al.* Whole-brain imaging with single-cell resolution using chemical cocktails and computational analysis. *Cell* **157**, 726–739 (2014).
63. Dodt, H.U. *et al.* Ultramicroscopy: three-dimensional visualization of neuronal networks in the whole mouse brain. *Nat. Methods* **4**, 331–336 (2007).
64. Schindelin, J. *et al.* Fiji: an open-source platform for biological-image analysis. *Nat. Methods* **9**, 676–682 (2012).
65. Schneider, C.A., Rasband, W.S. & Eliceiri, K.W. NIH image to ImageJ: 25 years of image analysis. *Nat. Methods* **9**, 671–675 (2012).