nature protocols

https://doi.org/10.1038/s41596-024-00957-5

Protocol

Check for updates

Modular segmentation, spatial analysis and visualization of volume electron microscopy datasets

Andreas Müller[1,2,3,9] ✉, Deborah Schmidt[4,9] ✉, Jan Philipp Albrecht[4,5], Lucas Rieckert[4], Maximilian Otto[4], Leticia Elizabeth Galicia Garcia[1,2,3,6], Gunar Fabig[7], Michele Solimena[1,2,3,6] & Martin Weigert[8] ✉

Abstract

Volume electron microscopy is the method of choice for the in situ interrogation of cellular ultrastructure at the nanometer scale, and with the increase in large raw image datasets generated, improving computational strategies for image segmentation and spatial analysis is necessary. Here we describe a practical and annotation-efficient pipeline for organelle-specific segmentation, spatial analysis and visualization of large volume electron microscopy datasets using freely available, user-friendly software tools that can be run on a single standard workstation. The procedures are aimed at researchers in the life sciences with modest computational expertise, who use volume electron microscopy and need to generate three-dimensional (3D) segmentation labels for different types of cell organelles while minimizing manual annotation efforts, to analyze the spatial interactions between organelle instances and to visualize the 3D segmentation results. We provide detailed guidelines for choosing well-suited segmentation tools for specific cell organelles, and to bridge compatibility issues between freely available open-source tools, we distribute the critical steps as easily installable Album solutions for deep learning segmentation, spatial analysis and 3D rendering. Our detailed description can serve as a reference for similar projects requiring particular strategies for single- or multiple-organelle analysis, which can be achieved with computational resources commonly available to single-user setups.

Key points

- This protocol provides a pipeline for analyzing volume electron microscopy datasets covering the preparation of raw data, the segmentation of specific organelles, their spatial analysis and three-dimensional visualization of the segmentation maps.
- The protocol demonstrates the use of tools such as Microscopy Image Browser, ilastik, Labkit and Album, which facilitates the installation of Python-based software (CSBDeep, CellSketch, StarDist, Blender and Jupyter notebooks).

Key reference

Müller, A. et al. *J. Cell Biol.* **220**, e202010039 (2021): https://doi.org/10.1083/jcb.202010039

[1]Molecular Diabetology, University Hospital and Faculty of Medicine Carl Gustav Carus, TU Dresden, Dresden, Germany. [2]Paul Langerhans Institute Dresden (PLID) of the Helmholtz Center Munich at the University Hospital Carl Gustav Carus and Faculty of Medicine of the TU Dresden, Dresden, Germany. [3]German Center for Diabetes Research, Neuherberg, Germany. [4]HELMHOLTZ IMAGING, Max Delbrück Center for Molecular Medicine (MDC) in the Helmholtz Association, Berlin, Germany. [5]Humboldt-Universität zu Berlin, Faculty of Mathematics and Natural Sciences, Berlin, Germany. [6]DFG Cluster of Excellence 'Physics of Life', TU Dresden, Dresden, Germany. [7]Experimental Center, Faculty of Medicine Carl Gustav Carus, Dresden, Dresden, Germany. [8]Institute of Bioengineering, School of Life Sciences, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland. [9]These authors contributed equally: Andreas Müller, Deborah Schmidt. ✉e-mail: andreas.mueller1@tu-dresden.de; deborah.schmidt@mdc-berlin.de; martin.weigert@epfl.ch

Introduction

For several decades, two-dimensional (2D) electron microscopy (EM) has been used to study the intricate details of subcellular components, contributing to key findings in cell biology and medicine. In recent years, three-dimensional (3D) volume electron microscopy (vEM) has established itself as a widely adopted new imaging modality, leading to a rapid proliferation of large raw image datasets of cells and tissues[1,2]. vEM includes methods such as serial section transmission electron microscopy (ssTEM), serial section electron tomography, serial block-face scanning electron microscopy (SBF–SEM) and focused ion beam scanning electron microscopy (FIB–SEM). The recent popularity of vEM has been made possible by major advances in sample preparation[3,4] as well as substantial improvements in stability and enhanced running time of instruments[5–8], resulting in impressive advances in connectomics[7,9,10] as well as in cell biology/biomedicine[11–19], and even enabling the reconstruction of whole organisms[20]. Moreover, many of these datasets have been made publicly available in open-access repositories/platforms such as EMPIAR[21], CEM500K[22] and OpenOrganelle[23]. The massive increase in data generation and accessibility has led to a surge in demand for analysis approaches. In particular, this requires segmentation maps of the desired structures throughout the volume followed by analysis of, i.e., volume fractions and/or investigation of spatial interactions. Until recently, a major part of the segmentation work had to be done manually within software packages such as IMOD[24], which can become extremely time consuming or even infeasible for larger datasets. Nevertheless, these approaches have led to impressive results[25,26].

The rise of machine learning methods has increased the possibility for automatic segmentation processes of a diverse set of organelle classes in large vEM images[20,27–32]. For example, Kaynig et al.[27] used random forest classifiers[33] to semi-automatically segment neurons from ssEM images. More recently, deep learning-based approaches based on U-Net[34] have shown impressive results on large-scale organelle segmentation from FIB–SEM volumes, as part of the Cell Organelle Segmentation in Electron Microscopy project[30], allowing pixel-wise segmentation for up to 35 cellular organelle classes. However, deep learning-based methods typically rely on labor-intensive manual ground truth annotation efforts, and often require computational resources that smaller laboratories cannot afford. Furthermore, the bioimaging community currently lacks protocols dealing with spatial analysis of 3D segmentation maps, such as the distribution of organelle distances, which prevents full exploitation of the raw data to obtain new biological insights. Finally, an effective visualization of the obtained volumetric segmentation maps is crucial for a holistic understanding of organelle distribution. At the same time, the fragmented landscape of visualization tools, that are often hard to learn, can be confusing for inexperienced users. In this protocol, we aim to address these issues by providing practical guidelines to perform the following key tasks on vEM images:

- Preparation of raw data (Procedure 1)
- Organelle-specific segmentation with a various set of methods (Procedure 2)
- Spatial analysis of the segmentation maps (Procedure 3)
- 3D visualization of segmentation maps (Procedure 4)

We present detailed steps for the segmentation of vEM datasets with user-friendly tools and give recommendations for which methods are most appropriate for each organelle class. We furthermore provide detailed steps to analyze the spatial interaction between the resulting 3D segmentation maps, thus enabling a comprehensive view of intracellular organelle interactions, which is essential for understanding the complex cellular ultrastructure. Finally, we present detailed instructions for generating 3D visualizations of the segmented organelles within their native cellular context and for creating accurate renderings for publication purposes and scientific outreach. Although these procedures were originally designed to be applied within a single project, we note that each procedure is modular and can be run independently given the availability of appropriate input data.

Development of the protocol

We initially developed this protocol (Fig. 1) to investigate the interaction of microtubules and organelles in quasi-isotropic FIB–SEM volumes of primary beta cells with a voxel size of 4 nm (ref. 11). Our aim was to analyze several complete beta cells in samples exposed to different metabolic stimuli (glucose concentrations). We focused on the organization of microtubules, their interaction with insulin secretory granules (SGs) and other organelles, as well as possible

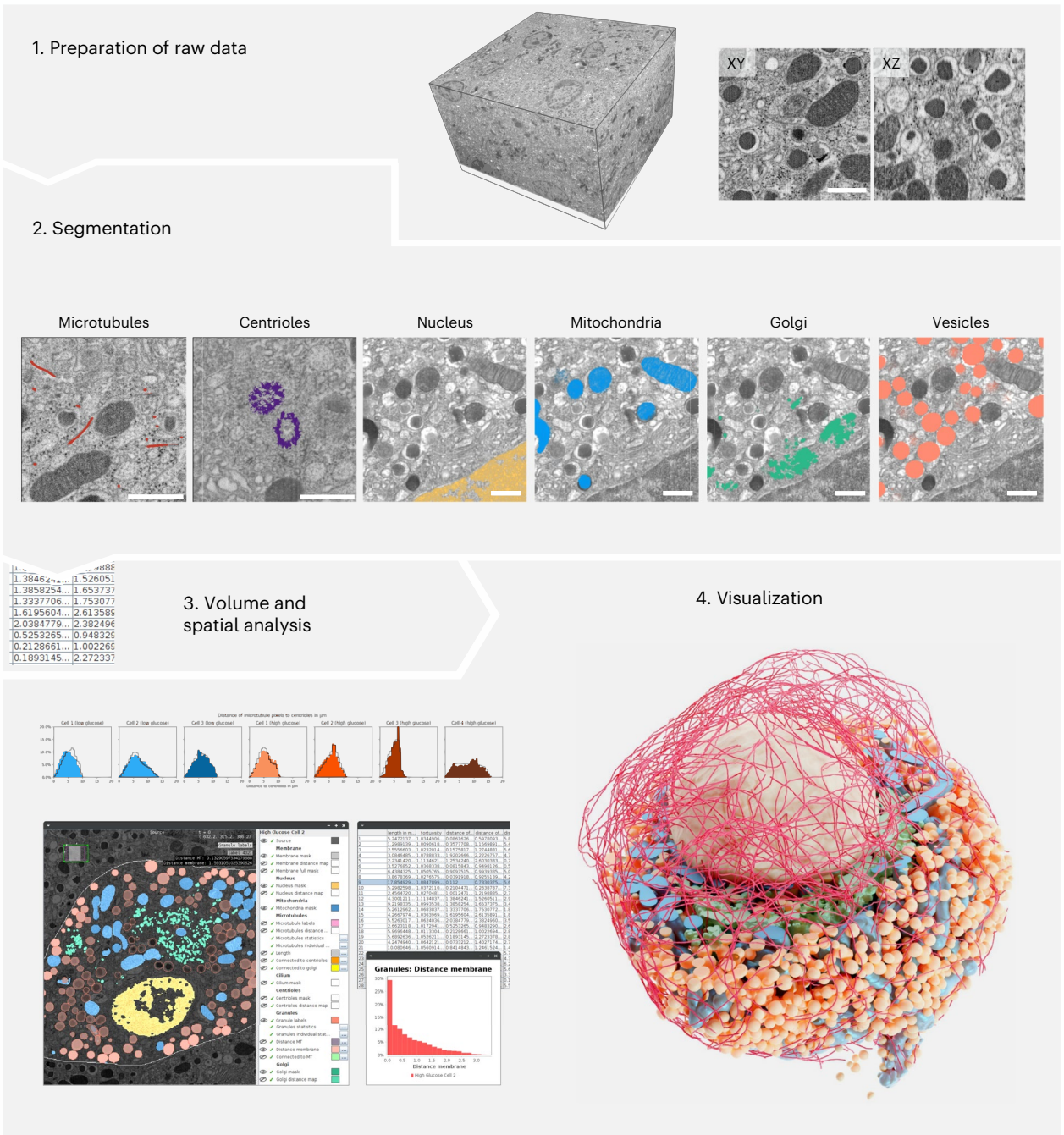

Fig. 1 | Overview of the protocol. Workflow starting with vEM data followed by organelle-specific segmentation, spatial analysis of the data and 3D rendering of the results. Analysis, plotting and 3D rendering steps can be performed via Album. Raw data and analysis image adapted from ref. 11, under a Creative Commons license CC BY 4.0.

changes upon stimulation with glucose. This required performing 3D segmentation of various organelles such as microtubules, insulin SGs, nuclei, centrioles, mitochondria, Golgi apparatus and the plasma membrane of each cell. These organelles strongly differ in many of their features: abundance in the cell, shape and electron contrast. Since some of them, such as the insulin SGs, amount to several thousand vesicles per cell, it became clear to us that full manual annotation is too time consuming and cumbersome. We also found that no single segmentation approach would perfectly match the specific features of all organelles simultaneously, and therefore chose dedicated methods that best suited a given organelle. Specifically, we required (i) mostly automated segmentation approaches that (ii) do not demand a large computational infrastructure and labor-intense prior annotation, can be (iii) implemented in user-friendly software and (iv) export file formats for segmentation results that can be easily shared between collaborators.

We used Microscopy Image Browser (MIB)[35] for manual segmentation of the plasma membrane and local thresholding of centrioles and nuclei. For more abundant and structurally conspicuous organelles such as mitochondria and (in a first attempt) insulin SGs we used autocontext classification[36] implemented in ilastik[37,38]. For more difficult tasks such as the Golgi apparatus, we turned to deep learning using a U-Net architecture[34] implemented in CSBDeep[39], whereas for refinement of insulin SG segmentation masks we used StarDist[40]. For full manual tracing of microtubules, we chose Knossos[41] since it allows memory-efficient annotation at the native pixel resolution, which is crucial for microtubules. Finally, manual curation of the segmentation results was performed in ilastik or the Labkit plugin[42] in Fiji[43].

We also found that a major hurdle for the proper analysis of 3D segmentation masks was the lack of protocols describing the analysis of complex spatial interactions. Analysis of organelle volumes is, e.g., possible in ilastik, IMOD or with the 3D object counter in Fiji[44]. However, high-resolution vEM segmentation masks enable the inspection of spatial interactions between organelles. This becomes especially interesting when investigating cells or tissues under different metabolic conditions, as in our case[11] and in other instances[12], or when comparing states in health and disease[45]. We therefore have devised a series of steps to perform these analyses, which helped us to unravel the connectivity of microtubules to centrioles and the Golgi apparatus as well as their close interaction with insulin SGs.

The final step was the 3D rendering of the segmentation results for publication purposes and scientific outreach. For 2D overlays of raw and segmentation data we used the Fiji plugin 3Dscript[46], which can also be used to generate 3D animations. For more engaging animations with shadows and special surfaces, we used Blender (https://www.blender.org/), which makes it necessary to convert the binary segmentation masks into meshes. This enables the generation of complex videos for public outreach.

Expertise needed to implement the protocol

This protocol is aimed at imaging scientists in biology and medicine with medium-level computational expertise who want to analyze vEM datasets of cells, tissues or small organisms. It is advantageous for users to be familiar with cellular ultrastructure as well as with the most widely distributed bioimage analysis tools, such as Fiji, ilastik and MIB. To help users to run Python/scripting-based tools in our workflow, we distribute them as Album solutions that streamline the process of dependency installation and allows users to execute workflows as installable graphical user interface (GUI) components.

Advantages, applications and comparison with other methods

Instead of relying on a single software, we provide a modular open-access workflow that enables users to pick out single methods for segmentation, analysis and rendering depending on their project. In contrast to multi-organelle segmentation approaches that use deep learning for all organelle classes, and thus rely on the creation of massive amounts of user-annotated ground truth for all organelles, our pipeline judiciously chooses organelle-specific segmentation methods to reduce unnecessary annotation effort. This strategy enables to achieve reasonable results early in the project for certain organelles, so that more time can be spent on difficult segmentation tasks. Nevertheless, the modularity of our approach allows users to still

incorporate their preferred segmentation tools. Integration of spatial analysis and 3D rendering into easily executable, interactive solutions with Album[47] allows users to best address complex organelle interactions within cells and to obtain novel biological insights from their vEM datasets. Furthermore, we provide a simple yet effective Blender workflow that enables the complete inner life of cells to be visualized in 3D that augments common perceptions derived from only 2D images.

The use case that we primarily address in this protocol is organelle segmentation in FIB–SEM volumes of beta cells. However, we designed the segmentation, analysis and rendering workflows in a way that allows the simple adaptation to other cell types and vEM modalities. Possible research questions include investigating the interaction of organelles with microtubules, contact sites between organelles such as mitochondria and the endoplasmic reticulum (ER), and changes in organelle morphology and interactions under different metabolic stimuli or disease conditions. We also aim to help scientists to decide which segmentation workflow suits their target structure best and to find a time- and labor-efficient strategy. We originally developed the protocol for analysing isotropic FIB–SEM volumes containing complete cells. However, most of the presented methods can also be applied to anisotropic datasets derived from ssTEM or SBF–SEM imaging or smaller volumes comprising only a few serial sections or single tomograms. Due to its modular design, our approach is complementary to other vEM segmentation and analysis tools (e.g., Amira, Empanada[48], workflows provided by ZeroCostDL4Mic[49], pretrained models of the BioImage Model Zoo[50]) as these can be integrated in our protocol at the appropriate steps. Therefore, our strategy can serve as a time- and labor-efficient framework for vEM analysis.

Limitations

Currently, we do not present strategies for all organelle classes such as ER, endosomes or lysosomes. Time-efficient tools to segment these structures as well as alternative programs, which could be implemented into our workflows, are described in the next paragraph. Additionally, the completely manual annotation of microtubules presented in our protocol is currently very time consuming and poses a bottleneck for our project. There are approaches using template matching[51] or deep learning[52] that might help to automate this task in the future. Furthermore, vesicle segmentation with StarDist should work similarly well for other large dense core vesicle types whose ultrastructure resembles SGs (e.g., chromaffin granules). For other, more distinct vesicle types (e.g., synaptic vesicles) alternative approaches, such as in ref. 53 might be more appropriate.

Our protocol was tested on quasi-isotropic FIB–SEM data, and therefore some of the segmentation tools described might work less well for anisotropic data modalities such as ssTEM or SBF–SEM. There it might be necessary to switch to tools developed specifically for this type of data, such as CDeep3M (ref. 54), which can be included into our workflows.

Finally, our current data format (TIFF) for exporting most of the segmentation results is suboptimal since it does not allow for very efficient saving and reading of the data. Hence switching to hierarchical data-formats (e.g., Hierarchical Data Format 5 (HDF5) (ref. 55), N5 (ref. 56) or ZARR[57]) will be beneficial in the future (with the caveat that not all segmentation tools currently support these formats). Therefore, we only used HDF5 for segmentation in ilastik and kept TIFF for the other segmentation tasks. As a result, our workflow might struggle for excessively large vEM datasets as used, e.g., in connectomics. However, we use a modern hierarchical data format (N5) for the spatial analysis and plan to integrate these data formats more consistently in the future. Finally, designing interactive workflows with interconnected components that fully operate on hierarchical data formats would be beneficial for improving the scalability of the protocol.

Alternative tools

The field of image segmentation is progressing rapidly, especially due to the rise of deep learning-based methods. However, to our knowledge, there are no freely available tools that combine various segmentation approaches as well as analysis and visualization workflows. In the following, we highlight alternative and complementary tools for segmentation of vEM

data that we believe show a high potential for facilitating this workflow, although they may require more advanced computational skills. Tools such as SuRVoS[58,59] aim to combine manual with machine-learning segmentation to facilitate image annotation. Pretrained models such as MitoNet[48] hold great potential for facilitating segmentation tasks by eliminating the requirement of the generation of training data. Empanada, the associated napari plugin for instant segmentation of mitochondria, provides rapid segmentation results. As an alternative to CSBDeep U-Net, segmentation can be performed with DeepMIB[60] as part of MIB or deep learning-based approaches such as ASEM[32]. Furthermore, there are various ways to reduce the annotation effort for ground truth generation e.g., such as described in CebraEM[61]. Also, approaches such as 'Etch a cell'[62] involving volunteers for ground truth generation have been implemented successfully, e.g., for segmentation of the nuclear envelope[31]. For spatial analysis of organelles, there are currently only a few tools available including Neuromorph add ons[63,64] for Blender as well as Barrios[65] developed for the investigation of spatial interactions within neuronal tissue.

Experimental design

In this section, we give some methodological background to the four main subtasks that we describe in the protocol: raw data preparation, organelle segmentation, spatial analysis and visualization (summarized in Table 1).

Organelle segmentation

At the beginning of each vEM analysis project, it is pivotal to decide which organelles need to be segmented and which kind of interactions need to be analyzed to solve the specific scientific question (Box 1 and Fig. 2). We recommend a series of tools for segmentation of different organelle types. However, using all of these in a single project is still time consuming and, in many cases, information of most of the cell's organelles may not be fully necessary. Before beginning to annotate structures, it is important to have an overview on different methods and to decide which strategy might fit best the organelle to be segmented. We list segmentation strategies according to organelle features (Fig. 2), which helps decide which method to use; Table 2 summarizes the specific practical advantages of the software packages we chose for segmentation.

Semi-automatic local thresholding and manual segmentation. For some organelle classes with low abundance (e.g., centrioles) segmentation via locally adjusted thresholding at each instance can already be sufficient and at the same time relatively quick to perform. Local thresholding works by manually segmenting the area very close to the target organelle followed by black-and-white thresholding within this defined area. We use this strategy for nuclei and

Table 1 | Overview of the procedures, their content and their mode of installation

Task	Steps	Tool	Installation
Preparation of raw data (Procedure 1)	Binning/cropping of EM data	Fiji	GUI
	Conversion into software-specific formats	Knossos cuber, Fiji	GUI
Segmentation (Procedure 2)	Local thresholding	MIB	GUI
	Random forest classifier	ilastik	GUI
	Deep learning/U-Net segmentation	CSBDeep	Album
	Shape-aware Deep learning	StarDist	Album
Spatial analysis (Procedure 3)	Creating a new CellSketch project accumulating segmentation results	CellSketch	Album
	Computation of spatial statistics	CellSketch	Album
	Visualization of spatial statistics	CellSketch	Album
	Plotting spatial statistics	Jupyter Notebook (Python)	Album
Visualization (Procedure 4)	Conversion of segmentation masks to meshes	VTK (Python)	Album
	Inspection of meshes	VTK (Python)	Album
	3D rendering	Blender	Album

BOX 1

How to choose a segmentation method

Different organelle classes pose different challenges to an automatic segmentation method: a method that accurately segments mitochondria might not be a suitable choice for delineating the Golgi apparatus, and vice versa. Knowing which segmentation method is best suited for each organelle class is thus critical for both the accuracy of the final results and to avoid time-consuming annotations in cases where they would not be required. In general, this choice depends on structural features of the organelle such as typical image contrast, organelle abundance and morphological complexity. Rare and contrast-rich organelles such as centrioles or nucleus, for example, often can be segmented with relatively simple tools that do not require manual annotation effort, such as local thresholding. On the other hand, abundant but well separated organelles, such as mitochondria, typically are best segmented with machine learning-based methods such as random forests or CNNs, which in turn require a certain amount of manual annotations. For this class of organelles, we recommend starting with random forest classifiers (such as provided by ilastik) that enable quickly judging whether more advanced methods such as a U-Net based semantic segmentation with CNNs are required. Finally, some organelle classes may present unique challenges, such as the extreme dense and crowded distribution of SGs that are hard to properly separate. For such situations, problem-specific applications might exist, such as the shape-aware method StarDist for the detection of crowded spherical objects. In Fig. 2 we summarize our choice for manual, semi-automated or machine learning-based segmentation tools, which we used in our FIB–SEM project[11]. This shall serve as an inspiration for any new segmentation endeavor.

centrioles, and chose MIB as a tool as it makes it possible to export segmentation results in various formats, such as an Amira AM file or TIFF. For other classes (e.g., plasma membrane) such simple approaches might fail to detect the structure of interest, and laborious manual segmentation might become necessary. Manual segmentation is basically done by tracing the outlines of the target organelle with a pen or brush tool in every *z*-section of the stack. It can be facilitated by interpolation in the *z*-dimension every five or ten slices. We again used MIB for manual segmentation of the plasma membranes of the cells.

Random forest segmentation. Random forest-based segmentation methods[33,66] use an ensemble of decision trees to classify each pixel based on a set of predefined image features. Random forest classifiers need only sparse (e.g., scribble) annotations and are fast to train, making

Organelle features	Example	Segmentation strategy
- Low occurence–small ROI - High contrast - Low complexity	Centrioles, nucleus	Manual + semi-automatic, local thresholding (MIB)
- High contrast - Low complexity - High occurence	Mitochondria	Sparse labeling, random forests (ilastik)
- High complexity - High convolution	Golgi, ER	Deep learning, U-Net (CSBDeep)
- High occurence - High densitiy	Vesicles	Shape-aware deep learning, (StarDist)
- Low contrast - Expert knowledge required - Precise segmentation required	Microtubules	Manual skeleton tracing (Knossos)

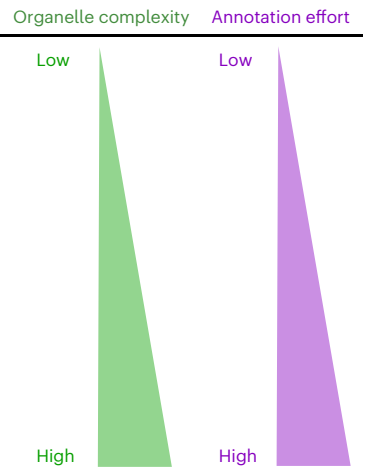

Fig. 2 | Overview of segmentation approaches on beta cell FIB–SEM datasets. Different organelles have characteristic features that should be considered when deciding on a segmentation approach. This enables the choice of time- and workload-efficient strategies.

Table 2 | Tools for specific segmentation strategies with their features that are beneficial for the segmentation task

Segmentation strategy	Tool	Important software features	Organelle	Procedure 2 steps
Local thresholding and manual segmentation	MIB	Local thresholding, interpolation, export to various formats	Membrane, nucleus	1–13
Random forests	ilastik	Fast learning based on sparse labeling, iterative improvement through the Autocontext feature	Mitochondria	14–32
Deep learning	CSBDeep	Able to segment highly convoluted and complex structures	Golgi	33–52
Shape-aware deep learning	StarDist	Prediction of polyhedral shape approximations for each object	SG	53–58
Manual skeleton tracing	Knossos	Memory-friendly data handling	Microtubules	59–64

them the classical machine-learning method of choice for initial segmentation experiments without the need for excessive manual annotation. Several software packages, such as MIB, Fiji (with the plugins trainable weka[67] or Labkit[42]) and ilastik provide random forest classifiers. Ilastik additionally implements a cascaded random forest classifier ('autocontext'), which can achieve better results than the simple pixel classification but includes two rounds of annotation and training. In the first round, the user needs to annotate many different organelles (e.g., nucleus, vesicles, mitochondria, ER, Golgi, ribosomes). After training, only the background and one organelle (e.g., mitochondria) are labeled. The second round of training will enable better results compared with a single-step pixel classification. We used this approach for segmentation of mitochondria and to generate ground truth data for insulin SGs and the Golgi apparatus.

Deep learning. Deep learning based on multilayered convolutional neural networks (CNNs) has become the most prevalent method for challenging image segmentation problems in the life sciences[68]. Commonly used CNN architectures can capture high- and low-resolution image semantics while containing millions of parameters that are learned from training data without the need for handcrafted features. Although this can result in very expressive and powerful segmentation models, the amount of training data required for good segmentation results is typically larger than for less expressive methods (e.g., random forests), and thus the manual annotation effort is often the bottleneck in those workflows (Box 2). The most widely used architecture for microscopy image segmentation is U-Net[34]. Moreover, network ensembles also have led to good results[54,69].

Shape-aware deep learning. For some organelle classes, it can be beneficial to exploit the structural priors they might possess, e.g., for SGs or other vesicles that are typically spherical in shape. In these cases, shape-aware deep learning methods that directly incorporate these priors in their prediction targets can be effective. Examples of these include StarDist[40], SplineDist[70], Cellpose[71] or local shape descriptors[72].

Skeleton tracing of filamentous structures. Components of the cytoskeleton such as microtubules are filamentous structures that can be represented as skeletons with a defined thickness. Since microtubules have an outer diameter of 25 nm, it was in our case necessary to use the full resolution of the dataset for annotation. Knossos uses a special file format, enabling the loading of only a small region surrounding the currently viewed area which makes it very memory efficient. Skeleton tracing is then performed by going through the volume and setting points throughout single microtubules from start to end, which are then connected by a line. These skeleton traces are then analyzed with a downstream procedure that we make accessible via Album.

Volume and spatial analysis

Analysis of the segmented data usually starts with calculating volumes and volume fractions of the different organelles with respect to the whole cell or raw volume. These calculations

BOX 2

How to generate ground truth

Machine learning-based segmentation methods (such as random forest and deep learning) require training data that consists of raw images and corresponding annotations (ground truth). For most random forest methods, sparse (scribble) annotations are sufficient for good segmentation results, whereas many deep learning-based segmentation methods require fully labeled ground truth images. As typically only a small part of the full raw images can be annotated with reasonable effort (e.g., 5–10 subregions of 128 × 128 × 128 pixels), the selection of which subregions to annotate is crucial for the performance of the trained model. These subregions should cover as best as possible the image variability seen in the full dataset. That means that you should not exclusively provide labelings of your organelle of interest, but also regions without it, such as extracellular space or regions rich in other organelles/structures. We also advise to label several small crops of the raw data and not a large chunk of the dataset. One should cut several small structurally diverse volumes ideally out of several raw datasets to ensure a good final segmentation result. To generate proper ground truth, a high level of expert knowledge is necessary. Classic EM textbooks and recent publications such as ref. 90 can help to decide how to distinguish organelles. As the generation of pixel-accurate annotation masks from scratch can be quite time intensive, we generally recommend creating preliminary ground truth with a fast and easy to use method first (e.g., ilastik) and then refine and curate them manually in a second step. This way, most of the annotation effort is spent on hard structures and facilitates the process of ground truth generation as well as enabling non-expert users to more quickly understand the structural features of the different organelles.

can be performed by several image analysis programs such as IMOD, Fiji, MIB or Amira. These data help to address the heterogeneity between cells and to investigate changes in organelle volumes, i.e., upon metabolic changes or in disease conditions. Precise segmentation masks also allow the analysis of distances between organelle classes either directly out of the binary masks or after creating 3D meshes. These approaches were first established for analyzing microtubule interactions in electron tomograms of mitotic spindles[73] and organelle interactions within the Golgi region of a beta cell line[74]. The latter study also introduced the comparison between observed and random distributions to make assumptions on regulated processes of organelle interaction, a method we have adopted for our original study. These kinds of calculations enable the investigation of complex interactions between organelles and the definition of subsets of an organelle based on its connectivity. Tools are partially implemented in IMOD, Amira and in the Neuromorph toolset for Blender[63,64]; however, they are not yet widely adopted. We generated distance transformations to calculate distances and connectivity between organelles with ImageJ2 (ref. 75) and ImgLib2 (ref. 76). Processing and plotting of these data were performed with Python-based tools[77–80]. In this protocol, we facilitate installation and execution of interactive workflow steps with Album.

Visualization

A successful 3D segmentation allows 3D rendering of the generated organelle masks to visualize all or a selection of organelles in their proper context (Box 3). Several tools are available that enable direct rendering of the original masks such as the 3D viewer[81] or 3Dscript[46] in Fiji or in Object Research Systems (ORS) Dragonfly. Dedicated 3D software tools such as Blender usually require conversion of the binary masks into 3D meshes. Before starting the 3D visualization, you should define a color scheme for your organelles with the help of online tools such as https://coolors.co/. Ideally, this should be colorblind friendly or at least structurally similar organelles such as different vesicle types should have distinct colors. You can check this with the Simulate Color Blindness plugin in Fiji.

BOX 3

How to effectively visualize results

As an integral part of basic research in medicine, biomedical image datasets can establish links to personal experiences of individuals with health or diseases. The 3D modeling and rendering software Blender is exceptionally well suited to tell the story behind a 3D dataset. Blender is extremely versatile within and beyond object material adjustment, illumination and animation. It is open source, freely available and fully scriptable, enabling us to simplify complex routines such as importing specific formats, adjusting the camera to frame all imported objects and assigning materials to the objects automatically. Scientific images are usually represented as pixel data—as a grid of points where each point has a specific value. To import such data into Blender, it first needs to be converted into a geometric structure by distinguishing between background and foreground pixels and generating a mesh of vertices along the border of the foreground, representing an object in 3D. Our protocol includes a routine for this purpose.

Materials

Equipment

Hardware requirements

As minimum requirements for raw data handling, segmentation and 3D visualization (Procedures 1, 2 and 4) we recommend a computer with:

1. At least 16 GB random-access memory (RAM)
2. An NVIDIA graphics processing unit (GPU) with at least 4 GB RAM
3. An up-to-date operating system (Windows 10/11, OSX 11+, Ubuntu 18+)
4. At least 500 GB of storage space

   For steps that involve the training of deep learning models (Procedure 2), we additionally require
5. An NVIDIA GPU with at least 8 GB memory
6. A working CUDA (≥10) installation (instructions are available at https://docs.nvidia.com/cuda/cuda-quick-start-guide/index.html#windows)

   For the original study we used
7. ThinkPad P52 with 32 GB RAM and an NVIDIA Quadro P3200 GPU with 6 GB VRAM with Windows 10
8. Workstation with 128 GB RAM and an NVIDIA RTX 2080 with 8 GB RAM with Ubuntu 18.04

   The analysis and visualization procedures were additionally tested on
9. Desktop computer with 32 GB RAM and an NVIDIA GeForce RTX 3060 with 12 GB VRAM with Windows 10
10. ThinkPad P14s with 48 GB RAM and an NVIDIA T500 with 4 GB VRAM with Ubuntu 20.04
11. MacBook Pro with OSX 12.3.1, with 32 GB RAM, M1 Max CPU/GPU

Datasets

12. Demo data (Table 3) to be used for individual steps of our protocol are available at https://zenodo.org/record/8114392

Software

Standalone (GUI-based) software

Standalone software packages used in this protocol that are available as cross-platform installable binaries and that can be executed via a GUI are listed in Table 4.

Album solutions

Several tools for segmentation, spatial analysis and visualization depend on custom python or java scripts that can be hard to install and execute (e.g., due to their dependencies). To facilitate

Table 3 | Demo data for the procedures

Task	Type of data	Link
Procedure 1		
FIB–SEM mouse 4 nm resolution	Raw data for binning	https://openorganelle.janelia.org/datasets/jrc_mus-pancreas-1, https://openorganelle.janelia.org/datasets/jrc_mus-pancreas-2
Procedure 2		
FIB–SEM mouse 16 nm resolution	Binned raw data for MIB/ilastik	https://zenodo.org/record/8114392/files/high_c1_source.tif
FIB–SEM mouse 16 nm resolution + masks	Training data for Golgi segmentation via CSBDeep/Album	https://zenodo.org/record/8114392/files/data_golgi.zip
FIB–SEM mouse 16 nm resolution + masks	Training data for granule segmentation via StarDist/Album	https://zenodo.org/record/8114392/files/data_granules.zip
Procedure 3		
Segmentation masks for multiple organelles	Input for CellSketch	https://zenodo.org/record/8114392/files/high_c1.zip
Segmentation masks for multiple organelles + distance transforms	Input for spatial analysis and visualization	https://zenodo.org/record/8114392/files/mycell.n5.zip
Procedure 4		
Input for rendering	n5	https://zenodo.org/record/8114392/files/mycell.n5.zip

installation and execution of these scripts, we wrap critical components into 'Album solutions'[47]. Album is a decentralized distribution platform for solutions to specific scientific problems. Each Album solution is a Python script including environment, custom installation routine descriptions, and specific metadata and runtime argument specifications. It works across platforms, tools and data domains and is designed to address limitations in reproducibility of scientific data software solutions and workflows, particularly when interactivity is needed. All routines in this protocol wrapped as Album solutions are bundled in an Album catalog and can be installed and launched from the GUI of Album. The Album documentation site (https://docs.album.solutions/) provides information on how to install and use Album solutions. If you run into any issues when running the Album installation wizard or during solution installation, refer to the Album troubleshooting page (https://docs.album.solutions/en/latest/faq.html). Software run via Album is listed in Table 5.

Table 4 | Standalone software used in this protocol

Tool	Version	Purpose	Link
Fiji	2.13.1	Software for bioimage visualization and analysis. Here, we use several plugins for our workflow	https://imagej.net/software/fiji/
Fiji plugins			
Ilastik	1.8.3	Plugin for converting TIFF to HDF5 files in Fiji and vice versa	https://github.com/ilastik/ilastik4ij
Labkit	0.3.11	Plugin for annotation and random forest classification. Here, we use it for annotation and cleaning of ground truth data	https://imagej.net/plugins/labkit/
ilastik	1.4.0	Software for image segmentation. Here, we use it for segmentation with random forests	https://www.ilastik.org/
MIB	2.8.4	Software for segmentation of EM data providing a large variety of tools. Here, we use it for manual and local thresholding segmentation of volume EM datasets	http://mib.helsinki.fi/
Knossos and Knossos cuber	v5.1	Software for opening and annotating large vEM datasets. Here, we use it for skeleton tracing of microtubules. KNOSSOS cuber converts volume EM files to cubed files readable by KNOSSOS	https://knossos.app/, https://github.com/knossos-project/knossos_cuber

Table 5 | Software installed and executed via Album

Tool	Version	Purpose	Link
CSBDeep	0.7.3	Toolbox for content-aware restoration of fluorescence microscopy images (CARE), based on deep learning via Keras and TensorFlow. Here, we use it for U-Net segmentation	https://github.com/CSBDeep/CSBDeep
StarDist	0.8.3	Object detection with star–convex shapes such as nuclei or vesicles	https://github.com/stardist/stardist/
CellSketch	0.2.1	3D toolbox for spatial analysis and visualization of single cell organelle masks and labels	https://github.com/betaseg/cellsketch
VTK	9.2.6	The Visualization Toolkit (VTK) is an open-source software for manipulating and displaying scientific data	https://vtk.org/
Blender	3.3.1	3D computer graphics software tool for rendering, visual effects, motion graphics, 3D modeling, etc.	https://www.blender.org/

CellSketch

CellSketch is a toolbox of procedures for collecting volumetric annotations (e.g., organelles) that are part of one object (e.g., a cell) in one project container, analyzing their spatial relationships and rendering them jointly. All imported data are stored in the N5 format as hierarchical blocks, enabling smooth interactive visualization of the project in BigDataViewer. Spatial analysis results between volumetric annotations are stored as tables. A routine for generating plots based on the analysis in a Jupyter Notebook is included. Finally, CellSketch provides automated routines of exporting project annotations as meshes, displaying them with VTK and rendering the project in Blender.

Procedure 1: preparation of raw data

▲ **CRITICAL** This protocol starts with an aligned vEM stack. Instructions on alignment can be found elsewhere[82,83]. The voxel dimensions of the vEM stack need to be properly calculated[84]. Also, it might be helpful to perform a denoising step on the dataset. There are various tools available, e.g., DenoisEM[85] or Noise2Void[86]. Before starting to segment, determine if you need to work on the full volume or if it is possible to cut certain regions of interest (ROIs), i.e., each ROI containing a cell (Fig. 3). Furthermore, it may not be necessary to use the full resolution of the original dataset to segment organelles such as mitochondria or nuclei. Binning by a factor of 2 or 4 might make sense as suggested in refs. 87,88 and our own work[11]. This will save a substantial amount of hard drive and memory space. We only used the full resolution for segmentation of microtubules. You should select and cut ROIs before binning.

Cutting ROIs containing complete cells

● **TIMING 10–30 min**

1. Install Fiji by navigating to https://imagej.net/software/fiji/downloads and choosing the download file according to your operating system.
2. Open the full resolution image stack in Fiji.
   ◆ **TROUBLESHOOTING**
3. Define the ROI in *xy* with the 'Rectangle' tool followed by ▷ Image ▷ Crop.
4. Afterwards, go through the resulting stack in *z* and define the start and end of the cell.
5. Cut the cell in *z* by ▷ Image ▷ Stacks ▷ Tools ▷ Slice Keeper and define 'First Slice' and 'Last Slice'.
6. Make sure to use 'Increment' 1 and press OK.
7. Save the resulting stack as TIFF file.

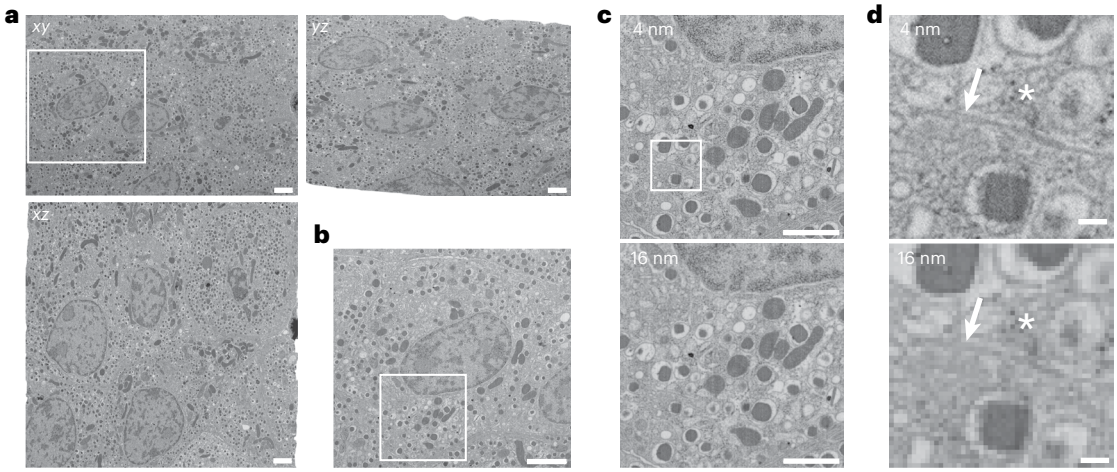

Fig. 3 | Preparation of raw data for segmentation. a, Full FIB–SEM volume of pancreatic islets containing several cells in *xy*, *xz* and *yz* views. Scale bars, 2 µm. **b**, The cell in the marked region of **a** is cut out in Fiji. Scale bar, 2 µm. **c**, Magnified view of boxed region of **b** at the original resolution (4 nm voxels) and after binning (16 nm voxels). Scale bars, 1 µm. **d**, Further magnifications to demonstrate the differences in resolution. Microtubules (arrow) and ribosomes (asterisk) are barely visible at 16 nm voxels. Scale bars, 100 nm.

Binning of subvolumes

● **TIMING 5 min**

8. Open the subvolume in Fiji
9. Go to ▷ Image ▷ Transform ▷ Bin and define *X*, *Y* and *Z* 'Shrink Factor'. You also need to choose the 'Binning Method' (we recommend Average). Press OK and save the result as a TIFF file. Alternatively, you can use 'binvol' as part of IMOD for binning (see Supplementary Information).

Preparation for Knossos

● **TIMING 10–30 min**

▲ **CRITICAL** For skeleton tracing in Knossos, you will need to convert your full resolution vEM dataset with Knossos cuber.

10. Install the program by following the installation instructions on https://github.com/knossos-project/knossos_cuber.
11. Navigate to the folder containing Knossos cuber.
12. Open the 'scripts' folder and run the file 'knossos_cuber_gui.exe'.
13. This opens the GUI where you can define the 'source directory', the 'target directory' and 'source format'.
    ◆ **TROUBLESHOOTING**
14. After running the program, you can open the result in Knossos.

Preparation for ilastik

● **TIMING 10 min**

▲ **CRITICAL** Ilastik can open various file formats. However, for optimal performance, you should convert your original file to HDF5. You can do this with the ilastik plugin in Fiji.

15. Open your vEM stack in Fiji.
16. Go to ▷ Plugins ▷ Ilastik ▷ Export to HDF5.
17. Define the 'Export Path' and press OK.

Procedure 2: segmentation

▲ **CRITICAL** We recommend getting acquainted with user-friendly tools such as MIB and ilastik when starting the segmentation tasks.

Local thresholding and manual segmentation with MIB

● **TIMING 45–120 min**

▲ **CRITICAL** Example data for this workflow can be downloaded from https://zenodo.org/record/8114392/files/high_c1_source.tif.

1. Install MIB. You will find installation instructions at http://mib.helsinki.fi/downloads.html. Install the standalone version for your operating system if you do not have a MATLAB license.
2. Open MIB.
3. Open your raw dataset by clicking the '…' button and navigate to the folder where the file is saved in. Press OK and click on the file in the left sidebar (Fig. 4a).
4. Navigate to your organelle of interest (e.g., centrioles).
5. With the 'Brush' tool, draw a line close to (for local threshold) or precisely on the organelle.
6. Fill the annotation by pressing the F key.
7. Annotate every fifth or tenth slice depending on the voxel size of the dataset. Press the I key for interpolating between annotated slices.
8. After finishing manual annotation, go to ▷ Selection .... ▷ Mask ▷ 3D ▷ Current stack(3D) ▷ Add.
9. Now the outline of your annotation will appear in magenta.
10. Switch from the 'Brush' tool to 'BW Thresholding'. Make sure to check the box 'Masked area' to only perform thresholding within the mask and check the '3D' box.
    ▲ **CRITICAL STEP** Change the high and low thresholds until you see a reasonable labeling of your target organelle in green.
11. Convert your labeling into a mask by going to ▷ Selection .... ▷ Mask ▷ 3D ▷ Current stack(3D) ▷ Replace. Export the mask as an AM or TIFF file by clicking ▷ Mask ▷ Save mask as.
12. Import the file into Fiji.
13. It might happen that you do not see the annotated area after import into Fiji. In this case, go to ▷ Process ▷ Binary ▷ Convert to Mask. Select 'Huang' for binarization and click OK.
    ▲ **CRITICAL STEP** For plasma membrane segmentation and for the nucleus (especially if you want to perform spatial analysis later), you should do a precise annotation without local thresholding. In these cases, export the mask as in Step 11 after labeling.
    ◆ **TROUBLESHOOTING**

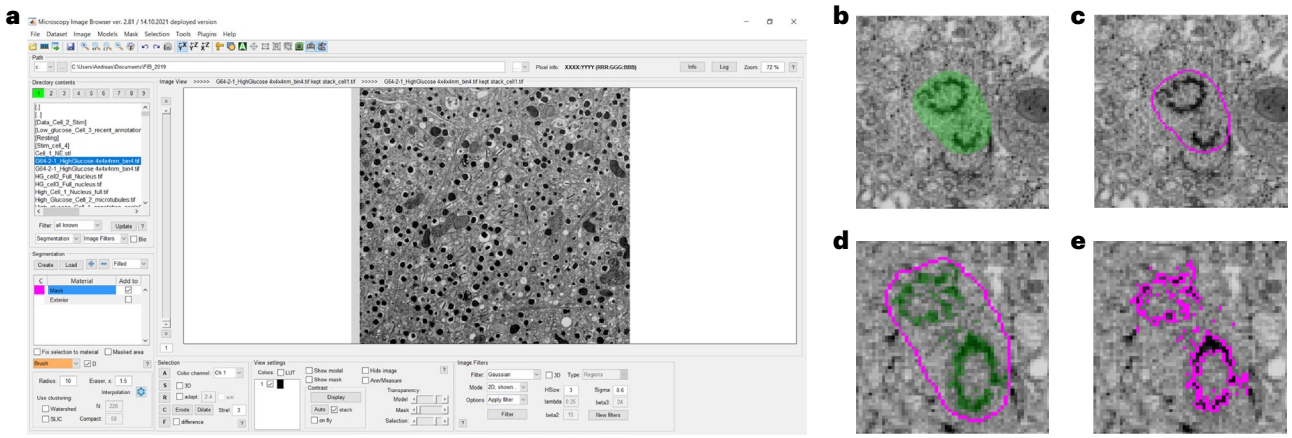

Fig. 4 | Local thresholding with MIB. a, Overview of the MIB window with raw data. **b**, *xy* view of the data with rough annotation surrounding the centrioles. **c**, Area of **b** after converting the annotation into a mask (magenta). **d**, Local thresholding (green) within the masked area. **e**, Conversion of the area to a mask that can be exported.

Random forests with ilastik

● **TIMING 24–48 h**

14. Install ilastik from https://www.ilastik.org/download.html according to your operating system.
15. Open ilastik and create a new project by clicking on 'Autocontext (2-stage)'. Define a file name for the project and save it.
16. Load the raw data in '1. Input Data' by clicking ▷ +Add New ▷ Add separate Image(s) and choose the HDF5 file you want to open. Ilastik will then display the file in *xy*, *yz* and *xz* view.
17. As the next step, select '2. Feature Selection' to select pixel features. Make sure to have 3D selected especially for isotropic data. For datasets with a huge difference between *x*, *y* and *z* values, it can make sense to select 2D features. More features will increase the likelihood for better results, but also require more computational resources. Press OK after selecting the features. For more information on feature selection, we recommend reading the ilastik documentation at https://www.ilastik.org/documentation/pixelclassification/pixelclassification#selecting-good-features.
18. Perform a first round of training (Fig. 5a). Click on '3. Training' and define labels for your organelles. Add new labels by clicking '+ Add Label', change colors of the labels by right click on the color and change label names by left double clicking.
19. After defining labels (in our case, vesicles, mitochondria, chromatin and background) perform sparse labeling of the structures with the 'Brush Cursor' tool for all labels.
    ▲ **CRITICAL STEP** In this first round, you should label more than two different organelles, ideally as many as you can discriminate.
20. Press 'Live Update' to see the probability map for each label (Fig. 5b). In this first step, this does not have to be perfect. However, you can improve the results by providing more annotations.
    ◆ **TROUBLESHOOTING**

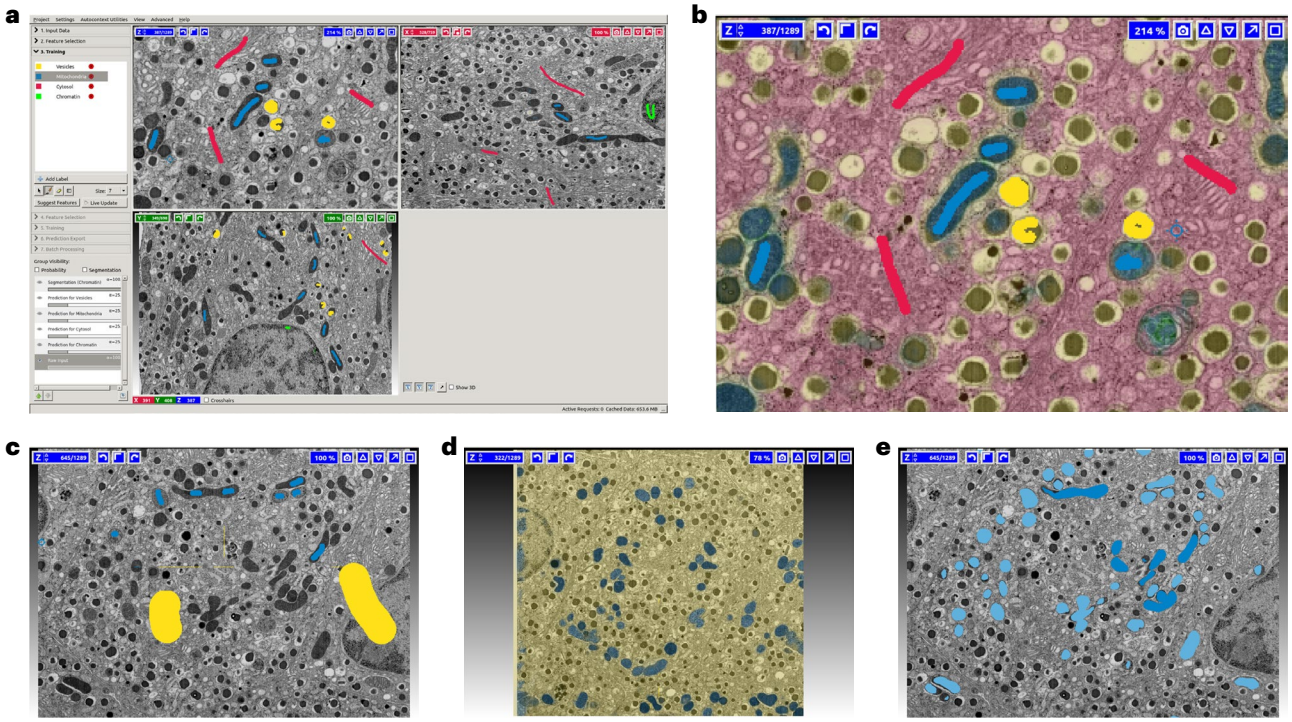

Fig. 5 | Autocontext segmentation in ilastik. a, Overview of the ilastik window with raw data and sparse annotations for four distinct structures (vesicles, mitochondria, cytosol, chromatin). **b**, *xy* view of the data after the first round of training with the different probability in the color code according to the labeling. **c**, *xy* view for the second round of annotation with labels for background (yellow) and mitochondria (blue). **d**, *xy* view after the second round of training with probabilities for background (yellow) and mitochondria (blue). **e**, Object detection of mitochondria (blue).

21. By clicking on 'Suggest Features', you can run an automated feature selection that can help to optimize the manual selection you did in the previous step. Make sure to click ▷ Project ▷ Save Project in between the steps.
22. If you are satisfied with the first round of training, you may continue to '4. Feature Selection' to select features for the second training.
23. After feature selection and clicking OK, proceed to '5. Training'. Define labels similar to the first training step. However, now you only provide annotation for your desired organelle (in our case, mitochondria) and background, which contains labels for all other organelles and the cytosol (Fig. 5c).
24. Press 'Live Update' and provide more annotations until you are satisfied with the result (Fig. 5d).
    ◆ **TROUBLESHOOTING**
25. If the probability map for your target organelle looks reasonable, you can proceed to '6. Prediction Export'. In the 'Export Settings', you can decide which file you wish to export. We will export 'Probabilities Stage 2' since we later want to perform object detection. Click on 'Choose Export Image Settings…', set the file format (HDF5) and give a file name.
26. Click OK and then click 'Export'.
    ◆ **TROUBLESHOOTING**
27. To generate binary masks needed for analysis you can either export the 'Simple Segmentation Stage 2' files after pixel classification or start a new project 'Object Classification [Inputs: Raw Data, Pixel Prediction Map]'.
28. Load raw data and the corresponding probability map.
29. Change thresholds until you are satisfied with the binary output.
30. In '3. Object Feature Selection', you can select the features you want to be computed.
31. In '4. Object Classification', you need to once label the organelle and press 'Live Update'. You will then see all mitochondria labeled (Fig. 5e) and can proceed to '5. Object Information Export'.
32. Export the 'Object Predictions' as HDF5.
33. You can import the HDF5 files into Fiji with the ilastik plugin by going to ▷ Plugins ▷ Ilastik ▷ Import HDF5 and select the file. It might be necessary to perform an additional binarization step, as described for importing files from MIB in Step 12. Afterwards you can save the file as TIFF.
    ◆ **TROUBLESHOOTING**

Deep learning for complex and/or convoluted organelles with CSBDeep

● **TIMING 12–24 h**

▲ **CRITICAL** For organelles that are highly convoluted and distributed throughout the whole cell, such as the Golgi apparatus or the ER, a supervised deep learning-based method is often the most efficient choice. As a semantic segmentation network, we use a U-Net as a robust and popular architecture for bioimages. For supervised deep learning, you need to provide precisely annotated ground truth (see box). We found that it can help to generate rough segmentation masks with Autocontext in ilastik (Steps 16–33) that can be cleaned in ilastik or Labkit similar to the data curation step at the end of Procedure 2, Steps 66–78.

▲ **CRITICAL** Demo data for this workflow can be found at https://zenodo.org/record/8114392/files/data_golgi.zip, consisting of nine FIB–SEM subvolumes of size 128 × 128 × 128 pixels and corresponding dense label masks of annotated Golgi apparatus (split into train and validation images/masks).

34. To generate ground truth from scratch with Labkit, open your raw dataset in Fiji.
35. Select square ROIs with the 'Rectangle' tool. Click on ▷ Edit ▷ Selection ▷ Specify and type in the identical x and y dimensions. We recommend 128 × 128 or 256 × 256 pixels. Press OK. Press ▷ Image ▷ Crop to crop the image in x and y.
36. To crop the resulting stack in z, go to ▷ Image ▷ Stacks ▷ Tools ▷ Slice Keeper and define the number of slices you want to keep (128 or 256). Make sure to set the increment to 1. Press OK and save the resulting stack.
37. Repeat this to generate ~20 subvolumes, ideally of various different raw volumes.

38. Open one of the volumes in Fiji and change the *z*-dimension to time [t] by clicking ▷ Image ▷ Properties and switch t- and z-values. Click OK and save the file.
39. Open the file in Labkit by clicking ▷ Plugins ▷ Labkit ▷ Open Current Image in Labkit.
40. Start annotation of your organelle of interest with the 'Draw' tool. You can navigate through the dataset in *z* with the left and right arrow keys. After carefully labeling all structures either with one foreground label or with several distinct labels, save your labeling as TIFF file.
41. As an alternative to Steps 33–39, you can generate ground truth with ilastik by preparing a first preliminary segmentation of the objects with Autocontext (as shown in Procedure 2, Steps 15–33) and export the resulting binary mask into Fiji (Fig. 6).
42. From the mask, prepare crops with dimensions of 128 × 128 × 128 pixels and if necessary, correct the labeling in Labkit (as described before).
43. Prepare 20 crops of corresponding raw images and labels masks and save them as TIFF files within two folders data/train/images and data/train/masks.
44. Use the ground truth label files and corresponding raw files to train a semantic segmentation model by executing the 'csbdeep_unet_train' solution provided via Album (expert users can may directly use a Jupyter Notebook for training without Album https://github.com/betaseg/protocol-notebooks/blob/main/unet/run_unet.ipynb. In this case, one can skip Steps 45–53.)

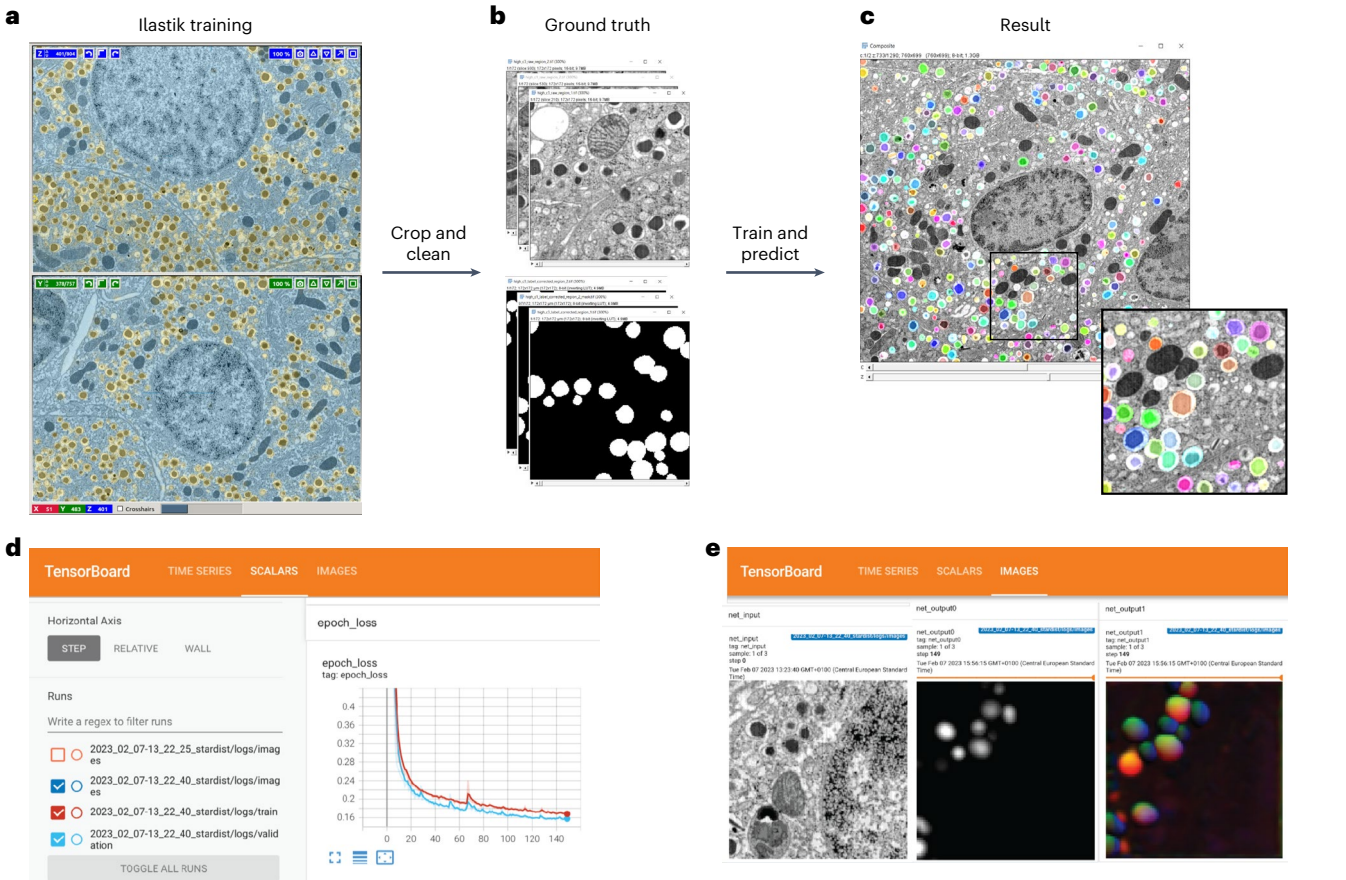

Fig. 6 | Segmentation of crowded insulin SGs with StarDist. a, Initial segmentation of SGs via the autocontext workflow in ilastik yields preliminary ground truth masks. **b**, Final ground truth masks are generated by selecting small crops and manually curating the labels. **c**, After training a StarDist model, it can be used to predict on new, full-sized stacks. **d**, Monitoring the StarDist model training via tensorboard. Shown are the loss curves for both training and validation, which should gradually decrease. **e**, The tensorboard will also show an example input slice (left) and the corresponding intermediate model outputs (center, object probability; right, distance maps), which both should get more and more refined during training.

45. Install Album using the install wizard (instructions here: https://docs.album.solutions/en/latest/installation-instructions.html), this generates a launcher for Album either on your desktop or in the list of available applications.
◆ **TROUBLESHOOTING**
46. After launching Album, click on ▷ catalogs in the bottom left corner.
47. Click ▷Add catalog.
48. Enter the following URL: https://github.com/betaseg/solutions and confirm. This is necessary for installing and running the specific solutions mentioned in the following steps. When running a solution which is not installed yet, you will be requested to install it first. You need to confirm an installation whenever one of the solutions is launched for the first time.
49. Prepare the data for running the U-Net training. To do so, split the raw data together with the generated ground truth into two folders named 'train' and 'val'. We recommend putting at least 20% of the available data in the 'val' folder. Each of these folders should have two subfolders: 'images' and 'masks', where the raw data and the annotated ground truth are stored, respectively (note: the demo data are already prepared in that way).
50. In the Album GUI install the 'CSBDeep U-Net Train' solution by clicking on ▷ run. In the following pop-up window click on ▷ install. This might take some time. Then, again click on ▷ run.
◆ **TROUBLESHOOTING**
51. Set the arguments for the solution. The only required argument is the root path of the prepared data from the previous step. Additional optional parameters are, for instance, the U-Net spatial pooling factor that should align with the anisotropy of the data (e.g., unet_pool_size=(2,2,2) for isotropic data or unet_pool_size =(2,4,4) for data with twice as large axial than lateral pixel size, as is the case here). Other parameters that might be adjusted are the 3D patchsize (128x128x128 by default) that is sampled during training and the batchsize (default 2). As output of the procedure, you obtain a trained model that can be used in downstream analysis.
◆ **TROUBLESHOOTING**
52. Click on ▷ run to train the model and monitor the training and validation loss in tensorboard. The training loss should gradually decrease, while the validation loss should plateau after 100–150 epochs. If the validation loss starts increasing heavily for later epochs, then the model overfits, which needs to be avoided. Note that model training is the most time-consuming part, that may take up to 8–14 h.
◆ **TROUBLESHOOTING**
53. In the Album GUI, install the 'CSBDeep U-Net Predict' solution. This might take some time. Then, click on ▷ run. Specify the root directory, input TIFF folder or file and an output directory, and start the inference by clicking again on ▷ run. The solution will automatically perform prediction on overlapping tiles to allow large stacks to be predicted with limited GPU memory. The prediction time of a stack of size 256^3 should be ~1 min.

Shape-aware deep learning for densely packed vesicles with StarDist

● **TIMING 12–36 h**

▲ **CRITICAL** For very abundant and densely packed star–convex (roundish) objects such as SGs or neuronal vesicles StarDist is a well-suited shape-aware segmentation method. StarDist directly predicts polyhedral shape approximations for each object and typically yields well-separated label maps even for very crowded scenarios.

▲ **CRITICAL** Demo data for this workflow can be found at https://zenodo.org/record/8114392/files/data_granules.zip consisting of five FIB–SEM subvolumes of size 128 × 128 × 128 pixels and corresponding dense label masks of annotated SGs.

▲ **CRITICAL** The StarDist training workflow in Steps 54–59 uses an Album solution, whereas for expert users, we provide a corresponding Jupyter Notebook that demonstrates how to train a 3D StarDist model on this data and how to use it to predict SG segmentation maps on new stacks here: https://github.com/betaseg/protocol-notebooks/blob/main/stardist/run_stardist.ipynb.

54. Prepare a first preliminary segmentation of the vesicles with Autocontext in ilastik (as shown in Procedure 2, Steps 15–33) and export the resulting binary mask into Fiji (Fig. 6). Out of this mask, prepare crops with dimensions of 128 × 128 × 128 pixels and, if necessary, correct the labeling in Labkit (as described in Procedure 2, Steps 66–71).
55. Prepare 20 crops of corresponding raw images and labels masks and save them as TIFF files within two folders data/train/images and data/train/masks.
56. If not yet done, install Album and the BetaSeg solutions catalog as described in Steps 45–48.
57. In the Album GUI, install the' StarDist Train' solution by clicking on ▷ run. In the following pop-up window, click on ▷ install. This might take some time. Then, again click on ▷ run. Use the crops you prepared as ground truth to train a StarDist 3D model with the help of the solution by specifying the root folder and the output folder. Note that we highly recommend using a workstation with an NVIDIA GPU for training if available. The process with a GPU will take up to 6–8 h.
    ◆ **TROUBLESHOOTING**
58. Monitor the model performance during training by observing the training/validation loss in tensorboard. The training loss should continuously decrease until the end of training, whereas the validation loss should similarly decrease for the first 50–100 epochs and then eventually plateau. Additionally, tensorboard will display the intermediate probability and distance predictions, which should converge to reasonable values.
    ◆ **TROUBLESHOOTING**
59. Use the trained model to predict on new (full) raw stacks using the 'StarDist Predict' Album solution. Use the Album GUI to install the solution. This might take some time. Then, again click on ▷ run. Specify the model base directory and model name you would, input TIFF folder or file, and an output directory and start the inference by clicking again on ▷ run. Depending on the size of the stack, the prediction step may take 0.2–1 h.

Manual skeleton tracing with Knossos

● **TIMING 1–2 weeks**

60. Install Knossos via https://knossos.app/.
61. Open Knossos and choose the dataset you want to work with (Fig. 7). Press OK.
62. Make sure that you have selected '6. Tracing Advanced' as work mode. Also open the cheat sheet by clicking on ▷ Help ▷ Cheatsheet.
63. Navigate through the dataset until you find a microtubule. Move to one of the ends of the microtubule and start to set nodes by a right mouse click.
64. After setting the second node, Knossos will connect the nodes. After reaching the end of the microtubule, press the C key to make a new tree (microtubule) and continue setting the nodes for the next microtubule.
65. Your annotation is automatically saved as an XML file, which is later used by Album for the spatial analysis.

Curation of segmentation results

● **TIMING 4–12 h**

▲ **CRITICAL** No segmentation result will be perfect, and we generally advise to proof-read each segmentation mask, ideally by an expert different from the original annotator. For mask curation, we recommend using either the Labkit plugin in Fiji or ilastik (but your preferred image analysis software might suffice).

66. For mask curation in Labkit. Open Fiji and load your dataset.
67. For making annotations in Labkit, it is recommended to change the *z*-dimension of your dataset to time [t]. You can do this by clicking on ▷ Image ▷ Properties and set Slices (z) to 1 and exchange the Frames (t) value with the number of *z*-slices. Press OK. Do the same also for the segmentation mask.
68. To load the raw data in Labkit, go to ▷ Plugins ▷ Labkit ▷ Open Current Image With Labkit.
69. After the Labkit window has opened and you see the raw data you can load the corresponding segmentation mask by clicking ▷ Labeling ▷ Open Labeling. Choose the file and click OK.

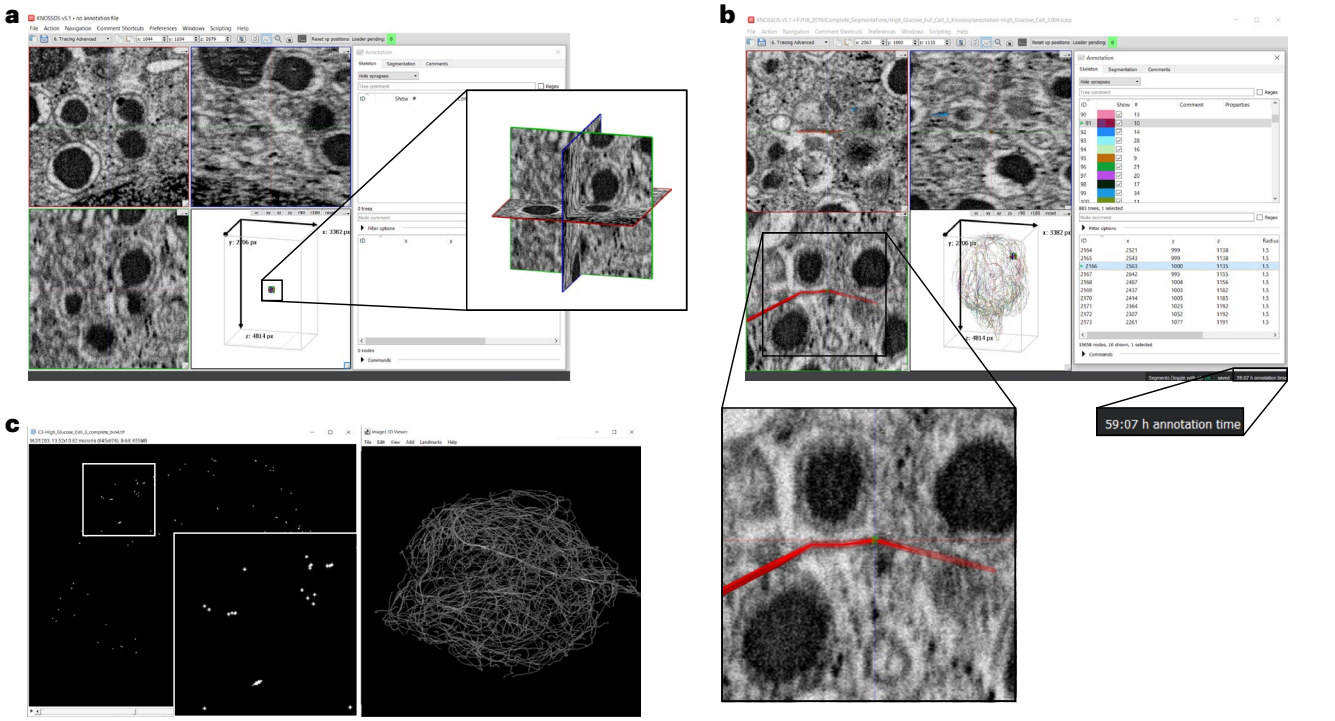

Fig. 7 | Skeleton tracing with Knossos. a, Overview of the Knossos window with raw data. The inset highlights the portion of the data that is currently loaded into memory. **b**, Knossos window including final skeleton trace of the cell. Left: the annotation window with single traces in different colors. Below: the nodes of the active trace. The insets highlight one trace with a node and the annotation time. **c**, Skeleton trace after conversion to binary mask in Fiji with a single slice through the stack on the left and 3D visualization in the 3D viewer on the right.

70. You can now navigate through the data with the left/right arrow keys and make corrections where needed. You can change the brush size and use the 'Draw' and 'Erase' tool.
71. Save your changes by clicking on ▷ Labeling ▷ Save Labeling. Make sure to save the file as TIFF.
72. For mask curation in ilastik. Open ilastik and create a new pixel classification project as shown before.
73. Open the raw data and proceed until Step '3. Training'.
74. Go to ▷ Advanced ▷ Labels ▷ Import Labels… and choose the corresponding segmentation file. Press OK.
75. You can now alter the labeling file with the 'Brush Cursor' and 'Eraser Cursor' tools. You should also give a few annotations for the background.
76. Click on 'Live Update' and run the prediction.
77. You can then export only the corrected label file in '4. Prediction Export' by choosing Labels as source.
78. After clicking export, you will save your desired segmentation file and the few labels for the background.

Procedure 3: volume and spatial analysis

▲ **CRITICAL** To perform the analysis of spatial interactions between organelles we use the Album solution CellSketch, which simplifies the installation and execution of steps spanning across multiple tools and programming languages. The most recent documentation of this

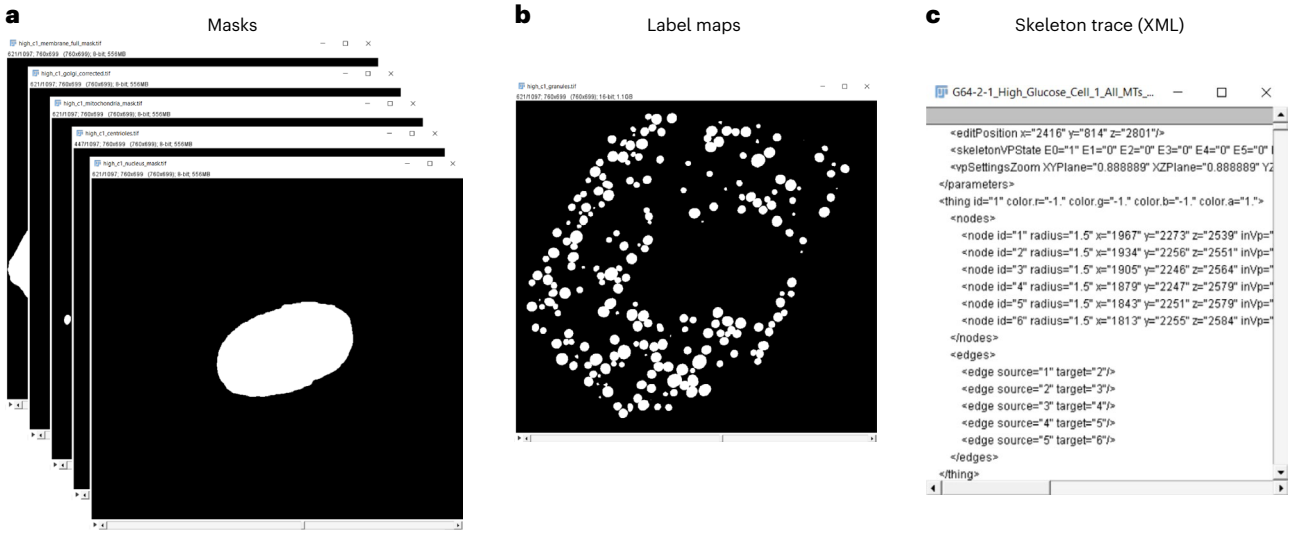

Fig. 8 | Types of data to be imported into the analysis project. a, 8 bit TIFF binary masks. **b**, 16 bit label maps. **c**, Skeleton traces XML files.

workflow can be found on https://github.com/betaseg/cellsketch, including the detailed command line calls for expert users. The following parts of the protocol show the full analysis workflow performed via the GUI of Album.

▲ **CRITICAL** Demo data for this workflow can be found at https://zenodo.org/record/8114392/files/high_c1.zip, which consists of the raw images and corresponding segmentation masks (as obtained by Procedure 2) for a mouse pancreatic cell. This dataset has an isotropic pixel size of 0.016 µm.

Creating a new CellSketch project

● **TIMING 20 min to 1 h**

1. To be able to follow the full analysis and segmentation protocol, make sure you have the following datasets available from Procedure 2 (Fig. 8):
   - Masks (e.g., TIFF, pixel value zero for background, pixel value 1 or 255 for foreground). According to Procedure 2, we expect the membrane mask to be filled. Furthermore, the pixel size must be isotropic, e.g., 0.016 µm for the demo data.
   - Label maps (e.g., TIFF, each object has the unique pixel value, 0 for background) of the vesicles. As before, the pixel size has to be isotropic.
   - The labeled microtubules are exported from Knossos as XML. As the Knossos analysis might have used raw images of higher resolution, ensure that the scale factors of the XML correspond to the lower pixel size of the masks. For example, for the demo data, the original microtubule segmentation was done on images with a pixel size of 4 nm × 4 nm × 3.4 nm, such that the scale factors need to be [X=0.25, Y=0.25, Z=0.2125] to result in the desired target isotropic pixel size of 0.016 µm.
2. Make sure you installed Album and added the BetaSeg catalog as described in Procedure 2, Steps 45–48. To visualize all segmentations and analysis values of individual organelle and microtubule instances in the same view, the data are transformed into a CellSketch project. CellSketch is a collection of solutions for displaying and analysing single cell components.
3. After launching Album use the search bar or scroll through your list of solutions to find and run the 'CellSketch: Create new project' solution. You then need to provide the following parameters:
   - parent: choose a folder where the project is going to be created
   - name: this name will represent the project, for the demo data you can call it high_c1
   - input: one dataset needs to be provided as the source image, for example, your raw dataset from the data acquisition. It will only be used for visualization purposes as the

background image. If none exists, one can simply choose one of the available masks. Any image file format that ImageJ/Fiji can open is expected to work
- pixel_to_um: the conversion factor from pixel units to µm units

Since spatial analysis and visualization are performed, the imported datasets need to be scaled to be isotropic. Use the scale_z parameter to provide a scaling factor for the *z* axis if needed.

4. A new folder in the specified parent folder with the ending .n5 will be created. Do not rename it.
5. Next, the project will be opened in the CellSketch viewer, displaying the raw dataset. The view supports arbitrary rotations in 3D. Hover over the right part of the display—an arrow on the right border will appear. Click on it to show the sidebar of the viewer.
◆ **TROUBLESHOOTING**
6. On the top of the right sidebar, there is a button called ▷ Add dataset. This will add several binary masks/label that represent different components of the organelle segmentations. A dataset has the following properties that you should set appropriately. Name: the name of the cell component. Will be used for displaying the component, in table columns and in file names i.e., when exporting meshes. Scale factor *X*/*Y*/*Z* for dataset: in case the dataset needs to be scaled to match the (isotropic) dimensions of the source datasets, these scaling parameters can be used. Color: the dataset will be displayed using this color—it can be later adjusted in the viewer by clicking the colored rectangle on the right side of the dataset name in the list of components in the right-side panel. Steps 7–10 below describe each of those components separately.
7. Add mask: mask datasets have the value 0 as background and 1 or 255 as foreground. They mark a component of the cell, such as the nucleus, without distinguishing between multiple entities of the same component.
8. Add labels: labels can be imported as label masks. Multiple entities of the same component type can be encoded by giving each object a unique pixel value whereas 0 is used for marking the background. A label mask does not support overlapping labels.
9. Add boundary: the boundary of the cell is a mask but plays a special role when analyzing the data. It describes the space which is available for components within the cell. Therefore, the boundary needs to be a filled mask, not just the membrane itself. CellSketch will automatically compute the outside border of this mask, add it as a cell component and call it 'membrane'.
10. Add filaments from Knossos (optional): in case you used Knossos to annotate filaments, they can be imported using this option. The following options can be set: analyze connection to filaments ends: check this box if your project contains filaments and if you want the connection of their ends to the label map or mask to be analyzed. Threshold to count filament ends as connected in µm: when calculating the distance between a mask or label with filaments, this threshold in micrometers is used to mark filaments as connected to this mask/label.
Note that since regularly exported Knossos files ignores the first *z*-slices of the dataset that do not contain annotations, this offset is internally added back depending on the number of *z*-slices of the source dataset of the project.
▲ **CRITICAL STEP** Adjust the microtubules scale values according to the correct voxel dimensions and binning factor of the raw data.
11. To delete an imported dataset, click on the three dots next to the bold name of the component in the side panel on the right and click ▷ Delete.
12. To display the project at a different time, use the search bar or scroll to the solution called 'CellSketch: Display data in BigDataViewer' in the Album interface and run it. Provide the newly created .n5 directory from the previous step as the project input parameter.

Spatial analysis

● **TIMING 30 min to 2 h**

▲ **CRITICAL** Spatial analysis can be performed by directly clicking the 'Analyze' button in the right-side panel of the viewer; however, we highly advise running the analysis from a separate solution without the viewer being opened. The analysis can be very memory consuming.

13. Run spatial analysis for your CellSketch project via the GUI by using the search bar or scrolling to the solution called 'CellSketch: Run spatial analysis'. Run the solution with the following parameters:
    - project: your CellSketch project, the directory ending with .n5.
    - connected_threshold_in_um: when analyzing how close two organelles need to be to be counted as connected, this is the threshold, provided in micrometers.

    ◆ **TROUBLESHOOTING**
14. All results of the analysis are stored into MY_PROJECT.n5/analysis. It will perform the following steps:
    - Distance maps for all imported components are computed. In a distance map, all pixel values represent the shortest distance of this pixel to a label or mask foreground
    - The mean, standard deviation (stdev) and median size of the labels of all label maps are computed and stored in PROJECT_NAME_LABELMAP_NAME.csv
    - The distance and connectivity of all labels of all label maps to all masks and other label maps are computed and stored in PROJECT_NAME_LABELMAP_NAME_individual.csv individually for each label. The number of connected versus the number of not connected labels are stored in PROJECT_NAME_LABELMAP_NAME.csv. This step includes the label maps of filaments - all filament pixels are considered, not just the filament ends
    - If filaments are present, the mean, standard deviation and median length and tortuosity of the filaments are stored in PROJECT_NAME_FILAMENTS_NAME.csv
    - If filaments are present, based on the parameters in the previous step, the distance between their ends and other labels/masks is computed and stored in PROJECT_NAME_FILAMENTS_NAME_individual.csv individually for each filament, and the number of connected versus not connected filaments are stored in PROJECT_NAME_FILAMENTS_NAME.csv

Visualizing spatial analysis in the CellSketch viewer

● **TIMING** 10 min

▲ **CRITICAL** Demo data for the plotting and visualization components of this protocol can be found at https://zenodo.org/record/8114392/files/mycell.n5.zip. The CellSketch project folder (high_c1.n5) already includes all analysis results (as obtained from Steps 13–14).

15. Launch the CellSketch viewer solution (Album ▷ CellSketch: Display data in BigDataViewer) to display the spatial analysis results.
16. Hover over the right part of the display—an arrow on the right border will appear—and click on it to show the sidebar of the viewer.
17. In the displayed list of all cell components, click on the eye symbols of the specific analysis component that should be visualized (Fig. 9). For example, displaying a distance map (i.e., the Golgi distance map) will show pixels further away from the respective target (in this case, the Golgi) in an opaquer white color. Displaying the attribute of a specific label, for example, the distance of SGs to the membrane, will color each label in respect to the value of the attribute for this specific object. All attributes of a specific label are also displayed as text in the top right corner of the viewer after clicking on it.

Exporting connectivity masks

● **TIMING** 10 min

▲ **CRITICAL** The CellSketch viewer can be used as a tool to generate TIFF masks of labels connected to specific masks. This can be useful for further analysis or visualization purposes.

18. Launch the CellSketch viewer solution (Album ▷ CellSketch: Display data in BigDataViewer) to display the spatial analysis results.
19. Hover over the right part of the display—an arrow on the right border will appear—click on it to show the sidebar of the viewer.
20. In the displayed list of all cell components, scroll to the entry of a label map dataset, for example, 'microtubules'. There should be connectivity entries listed below, for example, 'connected to centrioles' in case you added a centrioles mask.

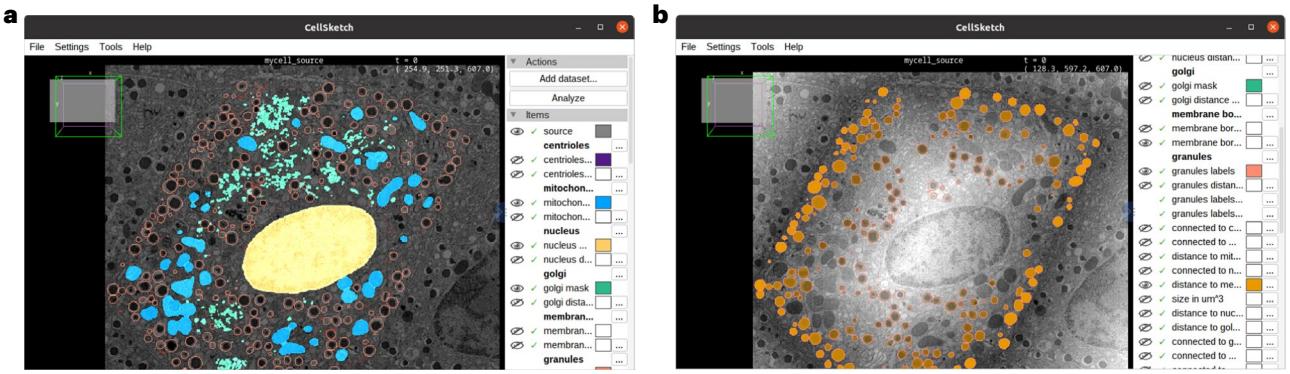

Fig. 9 | CellSketch viewer. a, The CellSketch viewer displaying the datasets imported into a new CellSketch project. **b**, The CellSketch viewer displaying the analysis results of a CellSketch project, in this case the distance of vesicles to the cell membrane and the distance map of the membrane.

21. Click on the three dots on the right side of a connectivity entry and click ▷ Export mask. Click ▷ Export inverted mask to export the labels which are not connected.
22. The masks are exported as TIFF. The progress bar in the ImageJ main panel, which opens alongside the CellSketch viewer, displays the export status. Once it is finished, the masks can be found in MY_PROJECT.n5/export.

Generating plots

● **TIMING 15 min to 1 h**

▲ **CRITICAL** We provide a Jupyter Notebook (https://github.com/betaseg/protocol-notebooks/blob/main/plots/run_plots.ipynb) showing how plots can be generated based on the analysis from the previous step. When running the notebook on the provided demo data, it should run without adjustments. To avoid installation of the Python dependencies, we distribute this workflow as an Album solution that makes it possible to run the notebook automatically in the correct environment, as follows:

23. After launching Album, use the search bar or scroll to the solution called 'CellSketch: Plot analysis results'. Run the solution with the following parameters:
    - project: your CellSketch project, the directory ending with .n5
    - output: the directory where plots generated in the notebook will be saved to (Fig. 10). This will also contain a copy of the notebook which can be further adjusted to generate more plots or to adjust the plots to a specific project
24. When running this command for the first time, the provided exemplary notebook will be executed and a copy of the notebook, including the result of the execution, will be stored

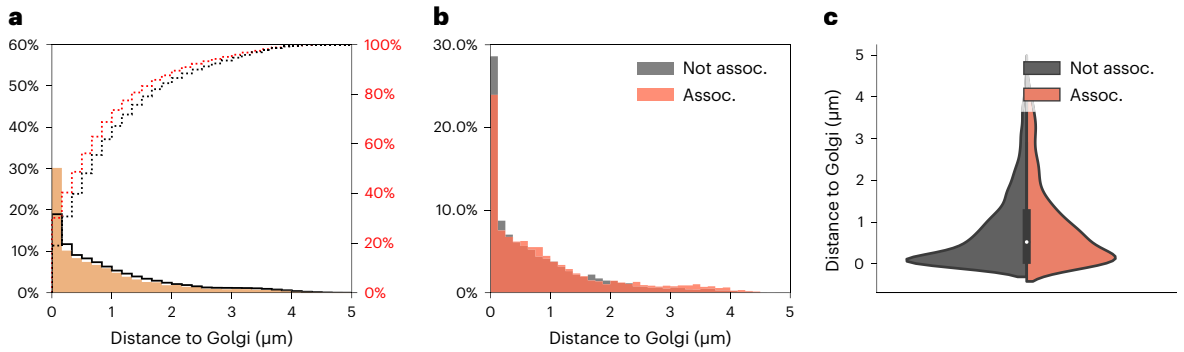

Fig. 10 | Plots generated from the spatial analysis results. a, Distribution of the distances of insulin SGs to the Golgi apparatus. The black solid line indicates a random distribution. Red dotted and black dotted lines represent actual and random cumulative distributions, respectively. **b**, Distribution of the distances of microtubule-associated (assoc.) and not-associated insulin SGs to the Golgi apparatus. **c**, Violin plot showing the distance of microtubule-associated and not-associated insulin SGs to the Golgi apparatus.

in the provided output directory. Afterwards, and when running the solution with an output directory that already contains the notebook, the solution will initiate a new Jupyter Notebook session and open the browser with the output Jupyter interface where one can click on plots.ipynb to adjust the notebook.

▲ **CRITICAL STEP** The notebook generates plots based on the datasets added to the projects programmatically, but it also includes a custom example accessing the analysis results of specific organelles explicitly. If the project has different cell component names than the exemplary dataset, this example will fail and the notebook will include messages that an exception occurred. Adjust the code in the notebook to generate plots based on the naming of the cell components in your project or delete the last cell.

Procedure 4: visualization of vEM segmentation data

▲ **CRITICAL** We focus on how to produce 3D renderings with Blender. Instructions for generating 2D overlays in 3D script and for 3D rendering in ORS Dragonfly can be found in the supplementary workflows. The steps for generating renderings in Blender are provided as Album solutions and integrated into the CellSketch workflow.

Visualization with VTK and Blender

● **TIMING 2–24 h**

▲ **CRITICAL** We provide Album solutions to automatically generate meshes from segmentations and labels masks, import the meshes into Blender, apply colored materials to each component and generate a scene that can be rendered in high quality. Before the full rendering in Blender, results can be quickly evaluated in the VTK.

▲ **CRITICAL** Demo data for this workflow is available at https://zenodo.org/record/8114392/files/mycell.n5.zip, which contains the CellSketch project folder high_c1.n5 with all analysis results obtained earlier.

▲ **CRITICAL** Make sure that you have installed Album and added the BetaSeg catalog as described in Procedure 2, Steps 45–48.

1. After launching Album, use the search bar or scroll to the solution called 'CellSketch: Export masks and labelmaps as meshes'. Run the solution with the following parameter:
   - project: your CellSketch project, the directory ending with .n5

   ▲ **CRITICAL STEP** The optional parameters 'include' and 'exclude' provide ways of choosing a subset of datasets. Only components containing the include string in their name will be exported, dataset names containing the exclude string will be ignored. Multiple inclusion strings can be provided by separating them by a comma, for example, mito,nucleus.
2. The meshes generated in the previous step were saved to MY_PROJECT.n5/export/meshes or the respective folder in your CellSketch project directory. Launch the solution called 'CellSketch: Display exported meshes' to visually inspect the exported meshes in VTK with your project as the input parameter (Fig. 11).
3. Either open Blender and import all generated STL files manually via ▷ File ▷ Import ▷ Stl or use the solution called 'CellSketch: Generate Blender scene from mesh files' to import all meshes jointly from a folder. The script will also adjust the viewport of the camera to ensure all meshes are visible and assign a glass material in the color as specified in the CellSketch project to each cell component. Run the solution with the following parameters:
   - project: your CellSketch project, the directory ending with .n5
   - output_blend: the path to where the Blender project file will be stored, useful for further adjustments of the scene

   ◆ **TROUBLESHOOTING**
4. You can manually assign or edit materials to each object by clicking on the ▷ Shading workspace in the top center area, selecting the object on the right, and adjusting or adding a material in the lower part of the user interface.

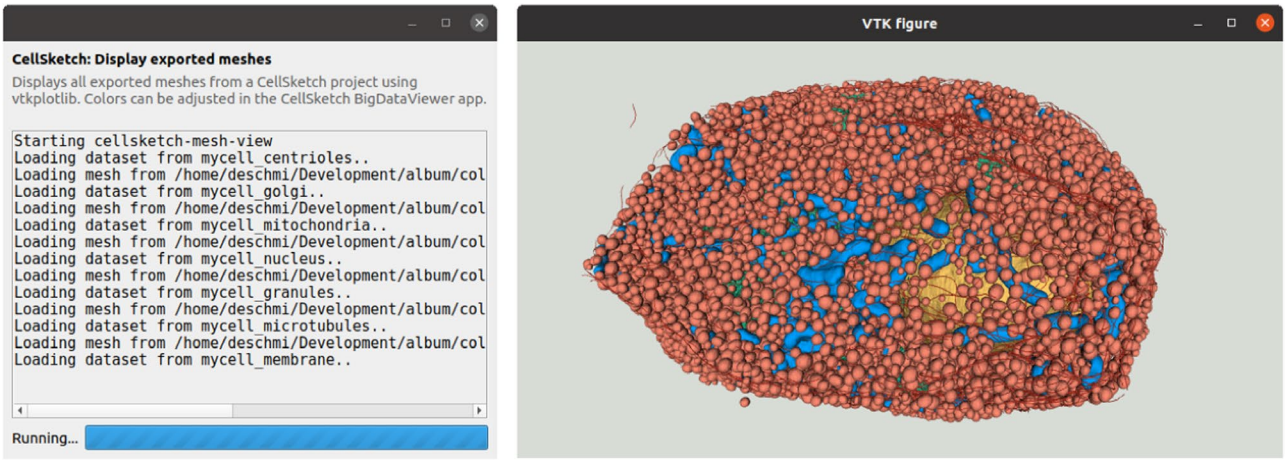

Fig. 11 | Exported meshes displayed in VTK. Cell components from a CellSketch project are automatically exported as meshes and displayed in VTK with a single solution call.

5. You can now open the Blender project stored at output_blend with Blender—either you already have Blender installed, then just open it there, or use the Album solution by launching Album, using the search bar, or scrolling to the solution called Launch Blender 3.3.1.
6. Click ▷ Render ▷ Render image to render the cell (Fig. 12c). Adjustments to the rendering can be done after selecting the ▷ Rendering workspace in the top centre area. Select Cycles in the right area as Render Engine for enhanced material rendering. GPU devices can be configured by clicking ▷ Preferences ▷ System ▷ Cycles Render Devices.

Check colors for color-blind friendliness

● **TIMING** 10 min

▲ **CRITICAL** In all the procedures above, one can define a color palette for the organelles, which should be used throughout the project (Fig. 12a). It is important that this color palette is adjusted to be color-blind friendly. You can check the palette file or one of the 3D renderings for color-blind friendliness by

7. In Fiji: go to Image ▷ Color ▷ Simulate Color Blindness.
8. Define the type of color blindness you want to check and press OK.
9. You will see how your colors will appear to people with the specific kind of color blindness.
10. Change your palette if the colors cannot be easily discriminated against.

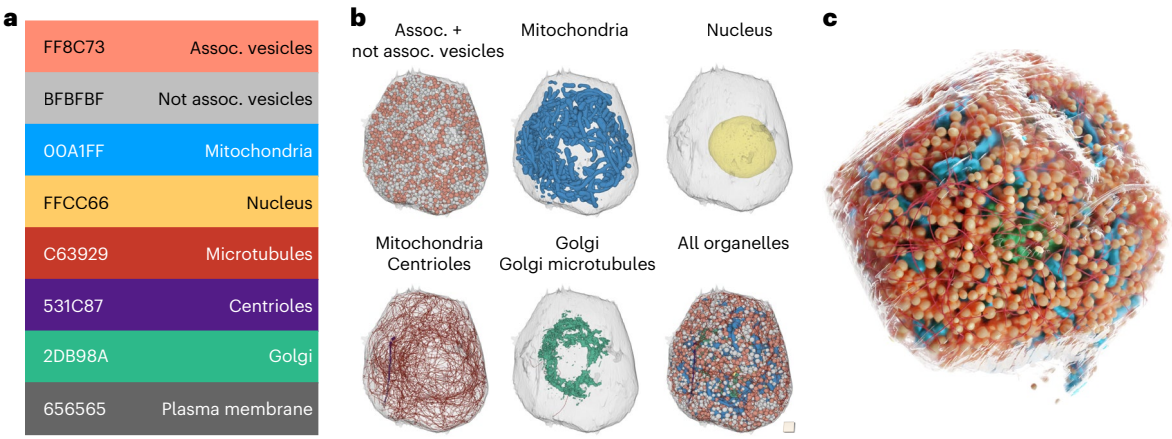

Fig. 12 | 3D visualization of vEM data. a, Color palette with HTML color codes and corresponding organelles. **b**, 3D renderings of organelles shown separately and as a complete cell in ORS Dragonfly. **c**, 3D rendering of the same cell as in **b** with Blender. Panel **b** adapted from ref. 11, under a Creative Commons license CC BY 4.0.

Troubleshooting

Troubleshooting advice can be found in Table 6.

Table 6 | Troubleshooting table

Step	Problem	Possible reasons	Solution
Procedure 1			
2	Full vEM volume cannot be opened	Raw data are too large for computer memory	Open file as virtual stack in Fiji or IMOD, or bin file before preparing crops
13	Conversion for Knossos does not work	Image stack is not present as single TIFF files	Convert stack into single TIFFs before conversion
Procedure 2			
13, 33	Imported segmentation mask is black in Fiji	You forgot to perform Object Detection in ilastik or you did not yet convert the file into a binary mask in Fiji	Perform the Object Detection step in ilastik by clicking once on the desired organelle and press 'Live Update'. After importing the mask into Fiji, go to Process ▷ Binary ▷ Convert to Mask and choose a method for binarization. Now your labels should be visible
20, 24	Ilastik probability map does not look reasonable	Not enough labels for training	Increase the number of labels or add more classes in the first Autocontext classification step
26	Export of ilastik probability map takes a very long time	Probability maps are very large files since they contain foreground and background masks in 32 bit	Let the export run overnight
45	The installation of Album via the Install Wizard asks for administrator permissions (Windows)	The wizard is running the Micromamba init call to add the Album Micromamba installation to the environment variables of the system in order to enable Album usage from the command line	You can ignore this request when using Album from the GUI. Also, refer to the Album troubleshooting page (https://docs.album.solutions/en/latest/faq.html)
	The installation of Album via the Install Wizard fails due to network issues	The wizard uses pip to download resources when installing Album	Adjust your network settings to allow access to the internet. Also, refer to the Album troubleshooting page (https://docs.album.solutions/en/latest/faq.html)
50, 53, 57, 59	Installation of Album solution fails	—	Refer to the Album troubleshooting page (https://docs.album.solutions/en/latest/faq.html)
	Solution installation fails due to missing git command	Some solutions may require git to download and install packages	Install git for your system (https://github.com/git-guides/install-git)
51, 57	GPU out of memory (before training)	The solution might expect a larger default GPU memory than available	Reduce the GPU limit by setting the 'limit_gpu_memory' value to the correct memory of your GPU (in MB)
52, 58	Validation loss is heavily increasing after a certain number of epochs (overfitting)	Indicates that training data are not sufficient or were wrongly annotated	Check your ground truth and correct it. Prepare more crops from diverse regions of the cell (also with negative examples) and label them precisely. Add more data augmentation
	GPU out of memory (during training)	The size of cropped training patches and/or batchsize is too high	Decrease the patchsize (e.g., to 96 × 96 × 96) or batchsize (e.g., to 2) parameter of the Album solution
Procedure 3			
5	Datasets are invisible/completely white	Datasets are displayed in the range of their minimal and maximal value. This sometimes is read incorrectly from the metadata	Navigate into the PROJECT.n5 directory. Continue to navigate into the directory of the incorrectly displayed dataset (i.e., PROJECT.N5/PROJECT_source.n5). Open the attributes.json file. It should look similar to this: {"min":0.0,"color":-8224126,"max":1.0,"dataType":"uint8","n5":"2.5.1","compression":{"type":"raw"},"blockSize":[64,64,64],"dimensions":[828,718,283]} Adjust or add the min and max values in there to the correct values and relaunch the CellSketch Viewer
13	Spatial analysis crashes or unexpectedly quits before finishing	The memory of the system is not sufficient to perform the analysis	Distance maps are the most memory-expensive to compute. In case the analysis process crashes because of memory issues, repeat the analysis process and select the 'skip existing distance maps' checkbox. It will not recompute already computed distance maps. This has to be unchecked whenever an existing dataset is deleted or changed. Otherwise, use a system with more RAM
Procedure 4			
3	Objects are not visible in Blender	Your objects are not aligned with the camera	Select one of your objects (i.e., membrane). Go to ▷ View ▷ Align View ▷ Align Active Camera to Selected

Timing

The information on timing in Procedure 1 is referring to one cell out of a large vEM stack with voxel dimensions of ~7,500 × 7,500 × 5,000 pixels. For Procedure 2 the timing is estimated for segmentation of one cell (~1/5 of the original stack) binned with factor 4 (except for microtubule segmentation, which was performed at the original resolution). The timing of processes 3 and 4 also refer to one cell of the original volume.

Procedure 1: preparation of raw data
Steps 1–7: 10–30 min
Steps 8–9: 5 min
Steps 10–14: 10–30 min
Steps 15–17: 10 min

Procedure 2: segmentation
Steps 1–13: 45–120 min
Steps 14–33: 24–48 h
Steps 34–53: 12–24 h
Steps 54–59: 12–36 h
Steps 60–65: 1–2 weeks
Steps 66–78: 4–12 h

Procedure 3: spatial analysis
Steps 1–12: 20 min to 1 h
Steps 13–14: 30 min to 2 h
Steps 15–17: 10 min
Steps 18–22: 10 min
Steps 23–24: 15 min to 1 h

Procedure 4: visualization
Steps 1–6: 2–24 h
Steps 7–10: 10 min

Anticipated results

This protocol provides modular steps for achieving the precise 3D segmentation of various organelle classes out of vEM datasets followed by spatial analysis and 3D visualization.

At the beginning of the protocol, we give advice on the preparation of the raw vEM data including binning and file conversion. After these first steps, the data will have the optimal size and file format(s) for the following segmentation tasks.

The user should be able to decide which segmentation strategy best matches their organelle of interest and will then able to perform these tasks in a time-efficient manner.

After successful segmentation and curation of the segmentation masks, the user will be able to perform volume analysis of the organelles as well as investigate spatial interactions between different organelle classes (Fig. 13a,d–f).

Finally, 3D rendering of the segmentation results can be applied to generate publication-ready figures and videos (Fig. 13b,c).

The steps for spatial analysis, plotting of the results and 3D rendering with Blender are implemented in solutions that can be run across platforms by using Album. This makes it possible to achieve publication-ready plots and renderings within a few hours after finalizing the segmentation tasks.

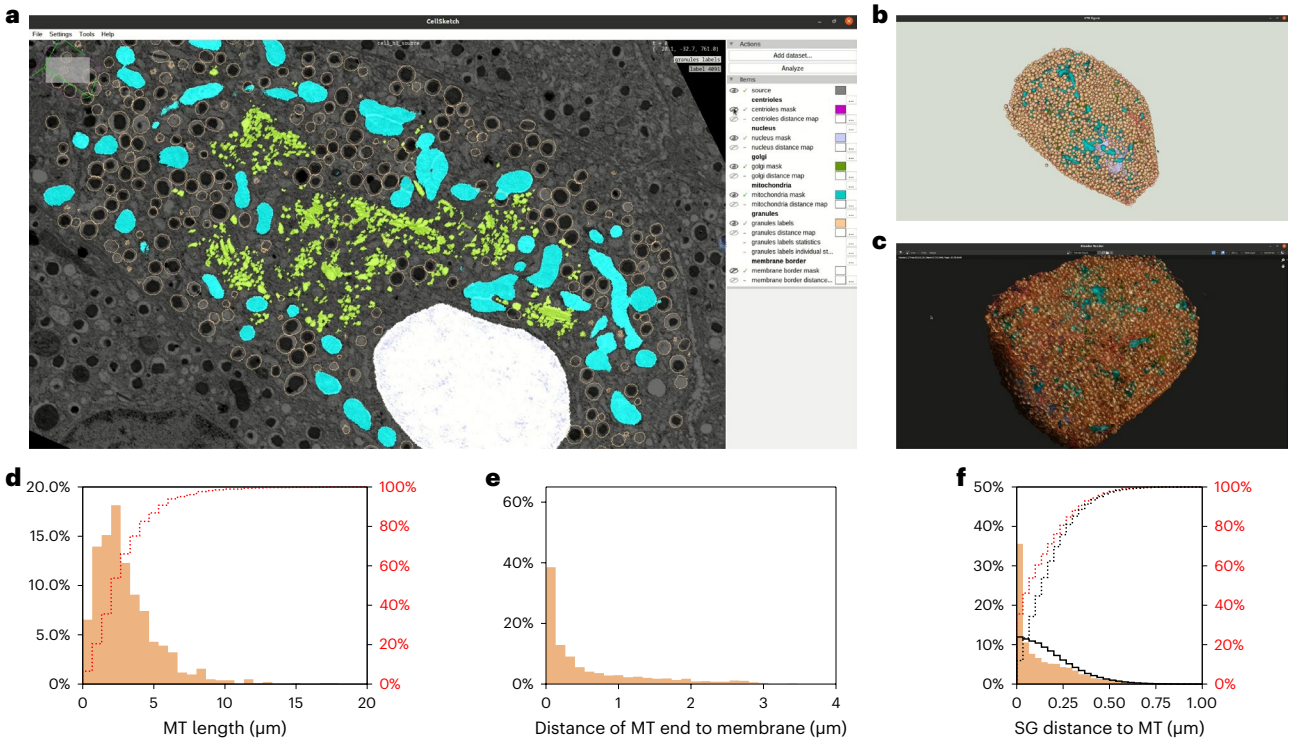

Fig. 13 | Results of segmentation, spatial analysis and 3D rendering of a vEM dataset. a, A screenshot of raw data and organelle segmentation masks loaded in CellSketch. **b**, 3D rendering in VTK of a complete cell after meshing with CellSketch. **c**, 3D rendering of the same cell in Blender. **d**, Plot depicting the microtubule (MT) lengths of the cell. The red dotted line indicates the cumulative data. **e**, Plot depicting the distance of MT ends to the plasma membrane of the cell. **f**, Distance of SGs to the plasma membrane. The black solid line indicates random distribution. The red dotted line indicates the cumulative data; black dotted line the cumulative random distribution.

The user can expect to get fundamental insights into their vEM data with information on the volume fractions, shapes and distribution of organelles within cells. When comparing different cell culture or disease conditions, the analysis enables addressing structural changes within cells in an unprecedented manner. In our project[11], we were able to evaluate heterogeneities between cells regardless of the culture conditions, by calculating volume fractions of organelles. We could find differences in microtubule lengths between metabolic stimuli—a key hint that metabolism influences ultrastructure. Spatial analysis enabled us to unravel the connectivity of microtubules in beta cells. They were largely noncentrosomal and surprisingly also not Golgi connected. Measuring the 3D association of insulin SGs with microtubules allowed us to categorize SGs in 'associated' and 'not associated', and the finding that microtubule–SG interaction preferably occurs in the cell periphery. Plotting spatial data against random distributions further allowed us to investigate possible regulated processes in the cell, which we could observe in the case of the enrichment of insulin SGs and microtubules close to the plasma membrane. In addition to the plotted data, 3D renderings helped to experience the data more intuitively and to appreciate better the crowding of the cytoplasm, the different cell and organelle shapes, as well as the basic organization of the microtubule scaffolds. Moreover, these procedures have recently been applied to reconstruct beta cell primary cilia out of vEM images[89]. In summary, our protocol provides a complete segmentation, analysis and visualization pipeline to fully embrace the complexity of modern vEM data.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

All raw datasets at their original resolution are available at OpenOrganelle (https://openorganelle.janelia.org/). Demo data of raw data, segmentation masks, deep learning training data and spatial analysis are available at https://zenodo.org/record/8114392. Further information on the demo data can be found in the Supplementary Information.

Code availability

The code for Album is available at https://gitlab.com/album-app/album. The code for the Album solutions is available at https://github.com/betaseg/solutions. The code for the CellSketch viewer is available at https://github.com/betaseg/cellsketch. The Jupyter Notebooks for demo workflows can be found at https://github.com/betaseg/protocol-notebooks.

Received: 7 March 2023; Accepted: 24 November 2023;
Published online: 29 February 2024

References

1. Peddie, C. J. & Collinson, L. M. Exploring the third dimension: volume electron microscopy comes of age. *Micron* **61**, 9–19 (2014).
2. Peddie, C. J. et al. Volume electron microscopy. *Nat. Rev. Methods Prim.* **2**, 51 (2022).
3. Hua, Y., Laserstein, P. & Helmstaedter, M. Large-volume en-bloc staining for electron microscopy-based connectomics. *Nat. Commun.* **6**, 7923 (2015).
4. Kievits, A. J., Lane, R., Carroll, E. C. & Hoogenboom, J. P. How innovations in methodology offer new prospects for volume electron microscopy. *J. Microsc.* **287**, 114–137 (2022).
5. Graham, B. J. et al. High-throughput transmission electron microscopy with automated serial sectioning. Preprint at *bioRxiv* https://doi.org/10.1101/657346 (2019).
6. Yin, W. et al. A petascale automated imaging pipeline for mapping neuronal circuits with high-throughput transmission electron microscopy. *Nat. Commun.* **11**, 4949 (2020).
7. Phelps, J. S. et al. Reconstruction of motor control circuits in adult Drosophila using automated transmission electron microscopy. *Cell* **184**, 759–774.e18 (2021).
8. Xu, C. S. et al. Enhanced FIB-SEM systems for large-volume 3D imaging. *eLife* **6**, e25916 (2017).
9. Motta, A. et al. Dense connectomic reconstruction in layer 4 of the somatosensory cortex. *Science* **366**, eaay3134 (2019).
10. Scheffer, L. K. et al. A connectome and analysis of the adult *Drosophila* central brain. *eLife* **9**, e57443 (2020).
11. Müller, A. et al. 3D FIB-SEM reconstruction of microtubule-organelle interaction in whole primary mouse β. *cells J. Cell Biol.* **220**, e202010039 (2021).
12. Parlakgül, G. et al. Regulation of liver subcellular architecture controls metabolic homeostasis. *Nature* **603**, 736–742 (2022).
13. Sheu, S.-H. et al. A serotonergic axon-cilium synapse drives nuclear signaling to alter chromatin accessibility. *Cell* **185**, 3390–3407.e18 (2022).
14. Weigel, A. V. et al. ER-to-Golgi protein delivery through an interwoven, tubular network extending from ER. *Cell* **184**, 2412–2429.e16 (2021).
15. Uwizeye, C. et al. Morphological bases of phytoplankton energy management and physiological responses unveiled by 3D subcellular imaging. *Nat. Commun.* **12**, 1049 (2021).
16. Musser, J. M. et al. Profiling cellular diversity in sponges informs animal cell type and nervous system evolution. *Science* **374**, 717–723 (2021).
17. Bharathan, N. K. et al. Architecture and dynamics of a desmosome–endoplasmic reticulum complex. *Nat. Cell Biol.* **25**, 823–835 (2023).
18. Malong, L. et al. Characterization of the structure and control of the blood-nerve barrier identifies avenues for therapeutic delivery. *Dev. Cell* **58**, 174–191.e8 (2023).
19. Cortese, M. et al. Integrative imaging reveals SARS-CoV-2-induced reshaping of subcellular morphologies. *Cell Host Microbe* **28**, 853–866.e5 (2020).
20. Vergara, H. M. et al. Whole-body integration of gene expression and single-cell morphology. *Cell* **184**, 4819–4837.e22 (2021).
21. Iudin, A., Korir, P. K., Salavert-Torres, J., Kleywegt, G. J. & Patwardhan, A. EMPIAR: a public archive for raw electron microscopy image data. *Nat. Methods* **13**, 387–388 (2016).
22. Conrad, R. & Narayan, K. CEM500K, a large-scale heterogeneous unlabeled cellular electron microscopy image dataset for deep learning. *eLife* **10**, e65894 (2021).
23. Xu, C. S. et al. An open-access volume electron microscopy atlas of whole cells and tissues. *Nature* **599**, 147–151 (2021).
24. Kremer, J. R., Mastronarde, D. N. & McIntosh, J. R. Computer visualization of three-dimensional image data using IMOD. *J. Struct. Biol.* **116**, 71–76 (1996).
25. Noske, A. B., Costin, A. J., Morgan, G. P. & Marsh, B. J. Expedited approaches to whole cell electron tomography and organelle mark-up in situ in high-pressure frozen pancreatic islets. *J. Struct. Biol.* **161**, 298–313 (2008).
26. Wu, Y. et al. Contacts between the endoplasmic reticulum and other membranes in neurons. *Proc. Natl Acad. Sci. USA* **114**, E4859–E4867 (2017).
27. Kaynig, V. et al. Large-scale automatic reconstruction of neuronal processes from electron microscopy images. *Med. Image Anal.* **22**, 77–88 (2015).
28. Dorkenwald, S. et al. Automated synaptic connectivity inference for volume electron microscopy. *Nat. Methods* **14**, 435–442 (2017).
29. Buhmann, J. et al. Automatic detection of synaptic partners in a whole-brain Drosophila electron microscopy data set. *Nat. Methods* **18**, 771–774 (2021).
30. Heinrich, L. et al. Whole-cell organelle segmentation in volume electron microscopy. *Nature* **599**, 141–146 (2021).
31. Spiers, H. et al. Deep learning for automatic segmentation of the nuclear envelope in electron microscopy data, trained with volunteer segmentations. *Traffic* **22**, 240–253 (2021).
32. Gallusser, B. et al. Deep neural network automated segmentation of cellular structures in volume electron microscopy. *J. Cell Biol.* **222**, e202208005 (2022).
33. Breiman, L. Random forests. *Mach. Learn.* **45**, 5–32 (2001).
34. Falk, T. et al. U-Net: deep learning for cell counting, detection, and morphometry. *Nat. Methods* **16**, 67–70 (2019).
35. Belevich, I., Joensuu, M., Kumar, D., Vihinen, H. & Jokitalo, E. Microscopy image browser: a platform for segmentation and analysis of multidimensional datasets. *PLoS Biol.* **14**, e1002340 (2016).
36. Tu, Z. & Bai, X. Auto-context and its application to high-level vision tasks and 3D brain image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **32**, 1744–1757 (2010).
37. Berg, S. et al. ilastik: interactive machine learning for (bio)image analysis. *Nat. Methods* **16**, 1226–1232 (2019).
38. Kreshuk, A. & Zhang, C. in *Computer Optimized Microscopy: Methods and Protocols* (eds Rebollo, E. & Bosch, M.) 449–463 (Springer, 2019).
39. Weigert, M. et al. Content-aware image restoration: pushing the limits of fluorescence microscopy. *Nat. Methods* **15**, 1090–1097 (2018).
40. Weigert, M., Schmidt, U., Haase, R., Sugawara, K. & Myers, G. Star-convex polyhedra for 3D object detection and segmentation in microscopy. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)* https://doi.org/10.1109/WACV45572.2020.9093435 (2020).
41. Helmstaedter, M., Briggman, K. L. & Denk, W. High-accuracy neurite reconstruction for high-throughput neuroanatomy. *Nat. Neurosci.* **14**, 1081–1088 (2011).
42. Arzt, M. et al. LABKIT: labeling and segmentation toolkit for big image data. *Front. Comput. Sci.* https://doi.org/10.3389/fcomp.2022.777728 (2022).
43. Schindelin, J. et al. Fiji: an open-source platform for biological-image analysis. *Nat. Methods* **9**, 676–682 (2012).
44. Bolte, S. & Cordelières, F. P. A guided tour into subcellular colocalization analysis in light microscopy. *J. Microsc.* **224**, 213–232 (2006).
45. Shrestha, N. et al. Integration of ER protein quality control mechanisms defines β-cell function and ER architecture. *J. Clin. Invest.* https://doi.org/10.1172/JCI163584 (2022).
46. Schmid, B. et al. 3Dscript: animating 3D/4D microscopy data using a natural-language-based syntax. *Nat. Methods* **16**, 278–280 (2019).
47. Albrecht, J. P., Schmidt, D. & Harrington, K. Album: a framework for scientific data processing with software solutions of heterogeneous tools. Preprint at https://doi.org/10.48550/arXiv.2110.00601 (2021).
48. Conrad, R. & Narayan, K. Instance segmentation of mitochondria in electron microscopy images with a generalist deep learning model trained on a diverse dataset. *Cell Syst.* **14**, 58–71.e5 (2023).

49. von Chamier, L. et al. Democratising deep learning for microscopy with ZeroCostDL4Mic. *Nat. Commun.* **12**, 2276 (2021).
50. Ouyang, W. et al. BioImage Model Zoo: a community-driven resource for accessible deep learning in bioimage analysis. Preprint at *bioRxiv* https://doi.org/10.1101/2022.06.07.495102 (2022).
51. Weber, B. et al. Automated tracing of microtubules in electron tomograms of plastic embedded samples of *Caenorhabditis elegans* embryos. *J. Struct. Biol.* **178**, 129–138 (2012).
52. Eckstein, N., Buhmann, J., Cook, M. & Funke, J. Microtubule tracking in electron microscopy volumes. In *Medical Image Computing and Computer Assisted Intervention (MICCAI) Part V* 99–108 (Springer, 2020).
53. Kaltdorf, K. V. et al. Automated classification of synaptic vesicles in electron tomograms of *C. elegans* using machine learning. *PLoS ONE* **13**, e0205348 (2018).
54. Haberl, M. G. et al. CDeep3M—plug-and-play cloud-based deep learning for image segmentation. *Nat. Methods* **15**, 677–680 (2018).
55. Koranne, S. in *Handbook of Open Source Tools* (ed. Koranne, S.) 191–200 (Springer, 2011).
56. Saalfeld, S. et al. saalfeldlab/n5: n5-2.5.1 https://doi.org/10.5281/zenodo.6578232 (2022).
57. Miles, A. et al. zarr-developers/zarr-python: v2.4.0 https://doi.org/10.5281/zenodo.3773450 (2020).
58. Luengo, I. et al. SuRVoS: super-region volume segmentation workbench. *J. Struct. Biol.* **198**, 43–53 (2017).
59. Pennington, A. et al. SuRVoS 2: accelerating annotation and segmentation for large volumetric bioimage workflows across modalities and scales. *Front. Cell Dev. Biol.* **10**, 842342 (2022).
60. Belevich, I. & Jokitalo, E. DeepMIB: user-friendly and open-source software for training of deep learning network for biological image segmentation. *PLoS Comput. Biol.* **17**, e1008374 (2021).
61. Hennies, J. et al. CebraEM: a practical workflow to segment cellular organelles in volume SEM datasets using a transferable CNN-based membrane prediction. Preprint at *bioRxiv* https://doi.org/10.1101/2023.04.06.535829 (2023).
62. Smith, P. et al. Online citizen science with the Zooniverse for analysis of biological volumetric data. *Histochem. Cell Biol.* https://doi.org/10.1007/s00418-023-02204-6 (2023).
63. Jorstad, A. et al. NeuroMorph: a toolset for the morphometric analysis and visualization of 3D models derived from electron microscopy image stacks. *Neuroinformatics* **13**, 83–92 (2015).
64. Jorstad, A., Blanc, J. & Knott, G. NeuroMorph: a software toolset for 3D analysis of neurite morphology and connectivity. *Front. Neuroanat.* **12**, 59 (2018).
65. Troidl, J. et al. Barrio: customizable spatial neighborhood analysis and comparison for nanoscale brain structures. *Comput. Graph. Forum* **41**, 183–194 (2022).
66. Schroff, F., Criminisi, A. & Zisserman, A. in *Procedings of the British Machine Vision Conference 2008* 54.1–54.10 (British Machine Vision Association, 2008).
67. Arganda-Carreras, I. et al. Trainable weka segmentation: a machine learning tool for microscopy pixel classification. *Bioinformatics* **33**, 2424–2426 (2017).
68. Hallou, A., Yevick, H. G., Dumitrascu, B. & Uhlmann, V. Deep learning for bioimage analysis in developmental biology. *Development* **148**, dev199616 (2021).
69. Shaga Devan, K., Kestler, H. A., Read, C. & Walther, P. Weighted average ensemble-based semantic segmentation in biological electron microscopy images. *Histochem. Cell Biol.* **158**, 447–462 (2022).
70. Mandal, S. & Uhlmann, V. Splinedist: automated cell segmentation with spline curves. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)* https://doi.org/10.1109/ISBI48211.2021.9433928 (IEEE, 2021).
71. Stringer, C., Wang, T., Michaelos, M. & Pachitariu, M. Cellpose: a generalist algorithm for cellular segmentation. *Nat. Methods* **18**, 100–106 (2021).
72. Sheridan, A. et al. Local shape descriptors for neuron segmentation. *Nat. Methods* **20**, 295–303 (2023).
73. McDonald, K. L., O'Toole, E. T., Mastronarde, D. N. & McIntosh, J. R. Kinetochore microtubules in PTK cells. *J. Cell Biol.* **118**, 369–383 (1992).
74. Marsh, B. J., Mastronarde, D. N., Buttle, K. F., Howell, K. E. & McIntosh, J. R. Organellar relationships in the Golgi region of the pancreatic beta cell line, HIT-T15, visualized by high resolution electron tomography. *Proc. Natl Acad. Sci. USA* **98**, 2399–2406 (2001).
75. Rueden, C. T. et al. ImageJ2: ImageJ for the next generation of scientific image data. *BMC Bioinforma.* **18**, 529 (2017).
76. Pietzsch, T., Preibisch, S., Tomančák, P. & Saalfeld, S. ImgLib2—generic image processing in Java. *Bioinformatics* **28**, 3009–3011 (2012).
77. Harris, C. R. et al. Array programming with NumPy. *Nature* **585**, 357–362 (2020).
78. Hunter, J. D. Matplotlib: a 2D graphics environment. *Comput. Sci. Eng.* **9**, 90–95 (2007).
79. Virtanen, P. et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* **17**, 261–272 (2020).
80. McKinney, W. Data structures for statistical computing in Python. In *Proc. of the 9th Python in Science Conference*. https://doi.org/10.25080/Majora-92bf1922-00a (2010).
81. Schmid, B., Schindelin, J., Cardona, A., Longair, M. & Heisenberg, M. A high-level 3D visualization API for Java and ImageJ. *BMC Bioinforma.* **11**, 274 (2010).
82. Cardona, A. et al. TrakEM2 software for neural circuit reconstruction. *PLoS ONE* **7**, e38011 (2012).
83. Hennies, J. et al. AMST: alignment to median smoothed template for focused ion beam scanning electron microscopy image stacks. *Sci. Rep.* **10**, 2004 (2020).
84. Hanslovsky, P., Bogovic, J. A. & Saalfeld, S. Image-based correction of continuous and discontinuous non-planar axial distortion in serial section microscopy. *Bioinformatics* **33**, 1379–1386 (2017).
85. Roels, J. et al. An interactive ImageJ plugin for semi-automated image denoising in electron microscopy. *Nat. Commun.* **11**, 771 (2020).
86. Krull, A., Buchholz, T.-O. & Jug, F. Noise2Void—learning denoising from single noisy images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 2129–2137 (2019).
87. Perez, A. J. et al. A workflow for the automatic segmentation of organelles in electron microscopy image stacks. *Front. Neuroanat.* **8**, 126 (2014).
88. Hoffman, D. P. et al. Correlative three-dimensional super-resolution and block-face electron microscopy of whole vitreously frozen cells. *Science* **367**, eaaz5357 (2020).
89. Müller, A. et al. Structure, interaction, and nervous connectivity of beta cell primary cilia. Preprint at *bioRxriv* https://doi.org/10.1101/2023.12.01.568979 (2024).
90. Park, G. et al. Amira annotation protocol. *protocols.io* https://www.protocols.io/view/amira-annotation-protocol-b834ryqw (2022).

Acknowledgements

We thank K. Pfriem for administrative assistance. We thank members of the PLID for valuable feedback. We thank S. Pang and C. Shan Xu (Yale University) as well as H. F. Hess (Janelia Research Campus) for FIB–SEM. We thank S. Kretschmar and T. Kurth from the Center for Molecular and Cellular Bioengineering Dresden (CMCB) for initial sample preparation. We thank all further authors of the original *Journal of Cell Biology* publication, J. Verner D'Costa, C. Münster (both PLID), and F. Jug (Human Technopole) for their support. This work was supported by the Electron Microscopy and Histology Facility, a Core Facility of the CMCB Technology Platform at TU Dresden. We thank B. Busselman (Universtiy of South Dakota) for testing Album installation. We thank the EM facility of the Max Planck Institute of Molecular Cell Biology and Genetics for their services. This work was supported with funds to M.S. from the German Center for Diabetes Research by the German Ministry for Education and Research (BMBF) and from the Innovative Medicines Initiative 2 Joint Undertaking under grant agreement no. 115881 (RHAPSODY) and 115797 (INNODIA). This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program and the European Federation of Pharmaceutical Industries and Associations (EFPIA). This work is further supported by the Swiss State Secretariat for Education, Research and Innovation under contract no. 16.0097-2. A.M. was the recipient of a MeDDrive grant from the Carl Gustav Carus Faculty of Medicine at TU Dresden. M.W. was supported by the ELISIR program of the École Polytechnique Fédérale de Lausanne School of Life Sciences and by generous funding from CARIGEST SA. D.S. and L.R. were funded by Helmholtz Imaging, a platform of the Helmholtz Information & Data Science Incubator.

Author contributions

A.M., D.S., M.S. and M.W. wrote the manuscript. D.S., J.P.A. and M.W. wrote the workflow implementations. L.R., M.O., G.F. and L.E.G.G. tested the workflows and provided feedback on the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41596-024-00957-5.

Correspondence and requests for materials should be addressed to Andreas Müller, Deborah Schmidt or Martin Weigert.

Peer review information *Nature Protocols* thanks Kedar Narayan and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

Related links

Key reference using this protocol

Müller, A. et al. *J. Cell Biol.* **220**, e202010039 (2021): https://doi.org/10.1083/jcb.202010039

nature portfolio

Corresponding author(s): Martin Weigert

Last updated by author(s): NP-P220514

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☐	☒	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☒	☐	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐	A description of all covariates tested
☒	☐	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☒	☐	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	All raw datasets at their original resolution are available at OpenOrganelle (https://openorganelle. janelia.org/). Demo data of raw data, segmentation masks, deep learning training data and spatial analysis are available at https://zenodo.org/record/8114392. Further information on the demo data can be found in the supplementary information.
Data analysis	The code for Album is available at https://gitlab.com/album-app/album. The code for the Album solutions is available at https://github.com/betaseg/solutions. The code for the CellSketch viewer is available at https://github.com/betaseg/cellsketch. The Jupyter notebooks for demo workflows can be found at https://github.com/betaseg/protocol-notebooks.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

All raw datasets at their original resolution are available at OpenOrganelle (https://openorganelle. janelia.org/). Demo data of raw data, segmentation masks, deep learning training data and spatial analysis are available at https://zenodo.org/record/8114392. Further information on the demo data can be found in the supplementary information.

Human research participants

Policy information about studies involving human research participants and Sex and Gender in Research.

Reporting on sex and gender	n/a
Population characteristics	n.a
Recruitment	n/a
Ethics oversight	n/a

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	This protocol is using publicly available data.
Data exclusions	n/a
Replication	n/a
Randomization	n/a
Blinding	n/a

Behavioural & social sciences study design

All studies must disclose on these points even when the disclosure is negative.

Study description	*Briefly describe the study type including whether data are quantitative, qualitative, or mixed-methods (e.g. qualitative cross-sectional, quantitative experimental, mixed-methods case study).*
Research sample	*State the research sample (e.g. Harvard university undergraduates, villagers in rural India) and provide relevant demographic information (e.g. age, sex) and indicate whether the sample is representative. Provide a rationale for the study sample chosen. For studies involving existing datasets, please describe the dataset and source.*
Sampling strategy	*Describe the sampling procedure (e.g. random, snowball, stratified, convenience). Describe the statistical methods that were used to predetermine sample size OR if no sample-size calculation was performed, describe how sample sizes were chosen and provide a*

	rationale for why these sample sizes are sufficient. For qualitative data, please indicate whether data saturation was considered, and what criteria were used to decide that no further sampling was needed.
Data collection	*Provide details about the data collection procedure, including the instruments or devices used to record the data (e.g. pen and paper, computer, eye tracker, video or audio equipment) whether anyone was present besides the participant(s) and the researcher, and whether the researcher was blind to experimental condition and/or the study hypothesis during data collection.*
Timing	*Indicate the start and stop dates of data collection. If there is a gap between collection periods, state the dates for each sample cohort.*
Data exclusions	*If no data were excluded from the analyses, state so OR if data were excluded, provide the exact number of exclusions and the rationale behind them, indicating whether exclusion criteria were pre-established.*
Non-participation	*State how many participants dropped out/declined participation and the reason(s) given OR provide response rate OR state that no participants dropped out/declined participation.*
Randomization	*If participants were not allocated into experimental groups, state so OR describe how participants were allocated to groups, and if allocation was not random, describe how covariates were controlled.*

Ecological, evolutionary & environmental sciences study design

All studies must disclose on these points even when the disclosure is negative.

Study description	*Briefly describe the study. For quantitative data include treatment factors and interactions, design structure (e.g. factorial, nested, hierarchical), nature and number of experimental units and replicates.*
Research sample	*Describe the research sample (e.g. a group of tagged Passer domesticus, all Stenocereus thurberi within Organ Pipe Cactus National Monument), and provide a rationale for the sample choice. When relevant, describe the organism taxa, source, sex, age range and any manipulations. State what population the sample is meant to represent when applicable. For studies involving existing datasets, describe the data and its source.*
Sampling strategy	*Note the sampling procedure. Describe the statistical methods that were used to predetermine sample size OR if no sample-size calculation was performed, describe how sample sizes were chosen and provide a rationale for why these sample sizes are sufficient.*
Data collection	*Describe the data collection procedure, including who recorded the data and how.*
Timing and spatial scale	*Indicate the start and stop dates of data collection, noting the frequency and periodicity of sampling and providing a rationale for these choices. If there is a gap between collection periods, state the dates for each sample cohort. Specify the spatial scale from which the data are taken*
Data exclusions	*If no data were excluded from the analyses, state so OR if data were excluded, describe the exclusions and the rationale behind them, indicating whether exclusion criteria were pre-established.*
Reproducibility	*Describe the measures taken to verify the reproducibility of experimental findings. For each experiment, note whether any attempts to repeat the experiment failed OR state that all attempts to repeat the experiment were successful.*
Randomization	*Describe how samples/organisms/participants were allocated into groups. If allocation was not random, describe how covariates were controlled. If this is not relevant to your study, explain why.*
Blinding	*Describe the extent of blinding used during data acquisition and analysis. If blinding was not possible, describe why OR explain why blinding was not relevant to your study.*

Did the study involve field work? ☐ Yes ☐ No

Field work, collection and transport

Field conditions	*Describe the study conditions for field work, providing relevant parameters (e.g. temperature, rainfall).*
Location	*State the location of the sampling or experiment, providing relevant parameters (e.g. latitude and longitude, elevation, water depth).*
Access & import/export	*Describe the efforts you have made to access habitats and to collect and import/export your samples in a responsible manner and in compliance with local, national and international laws, noting any permits that were obtained (give the name of the issuing authority, the date of issue, and any identifying information).*
Disturbance	*Describe any disturbance caused by the study and how it was minimized.*

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☐	☒ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging

Antibodies

Antibodies used: *Describe all antibodies used in the study; as applicable, provide supplier name, catalog number, clone name, and lot number.*

Validation: *Describe the validation of each primary antibody for the species and application, noting any validation statements on the manufacturer's website, relevant citations, antibody profiles in online databases, or data provided in the manuscript.*

Eukaryotic cell lines

Policy information about cell lines and Sex and Gender in Research

Cell line source(s): *State the source of each cell line used and the sex of all primary cell lines and cells derived from human participants or vertebrate models.*

Authentication: *Describe the authentication procedures for each cell line used OR declare that none of the cell lines used were authenticated.*

Mycoplasma contamination: *Confirm that all cell lines tested negative for mycoplasma contamination OR describe the results of the testing for mycoplasma contamination OR declare that the cell lines were not tested for mycoplasma contamination.*

Commonly misidentified lines (See ICLAC register): *Name any commonly misidentified cell lines used in the study and provide a rationale for their use.*

Palaeontology and Archaeology

Specimen provenance: *Provide provenance information for specimens and describe permits that were obtained for the work (including the name of the issuing authority, the date of issue, and any identifying information). Permits should encompass collection and, where applicable, export.*

Specimen deposition: *Indicate where the specimens have been deposited to permit free access by other researchers.*

Dating methods: *If new dates are provided, describe how they were obtained (e.g. collection, storage, sample pretreatment and measurement), where they were obtained (i.e. lab name), the calibration program and the protocol for quality assurance OR state that no new dates are provided.*

☐ Tick this box to confirm that the raw and calibrated dates are available in the paper or in Supplementary Information.

Ethics oversight: *Identify the organization(s) that approved or provided guidance on the study protocol, OR state that no ethical approval or guidance was required and explain why not.*

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Animals and other research organisms

Policy information about studies involving animals; ARRIVE guidelines recommended for reporting animal research, and Sex and Gender in Research

Laboratory animals: For mouse data see Müller et al. "3D FIB-SEM reconstruction of microtubule–organelle interaction in whole primary mouse β cells." Journal of Cell Biology 220.2 (2021). https://doi.org/10.1083/jcb.202010039

Wild animals	n/a
Reporting on sex	n/a
Field-collected samples	n/a
Ethics oversight	n/a

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Clinical data

Policy information about clinical studies

All manuscripts should comply with the ICMJE guidelines for publication of clinical research and a completed CONSORT checklist must be included with all submissions.

Clinical trial registration	*Provide the trial registration number from ClinicalTrials.gov or an equivalent agency.*
Study protocol	*Note where the full trial protocol can be accessed OR if not available, explain why.*
Data collection	*Describe the settings and locales of data collection, noting the time periods of recruitment and data collection.*
Outcomes	*Describe how you pre-defined primary and secondary outcome measures and how you assessed these measures.*

Dual use research of concern

Policy information about dual use research of concern

Hazards

Could the accidental, deliberate or reckless misuse of agents or technologies generated in the work, or the application of information presented in the manuscript, pose a threat to:

No	Yes	
☐	☐	Public health
☐	☐	National security
☐	☐	Crops and/or livestock
☐	☐	Ecosystems
☐	☐	Any other significant area

Experiments of concern

Does the work involve any of these experiments of concern:

No	Yes	
☐	☐	Demonstrate how to render a vaccine ineffective
☐	☐	Confer resistance to therapeutically useful antibiotics or antiviral agents
☐	☐	Enhance the virulence of a pathogen or render a nonpathogen virulent
☐	☐	Increase transmissibility of a pathogen
☐	☐	Alter the host range of a pathogen
☐	☐	Enable evasion of diagnostic/detection modalities
☐	☐	Enable the weaponization of a biological agent or toxin
☐	☐	Any other potentially harmful combination of experiments and agents

ChIP-seq

Data deposition

☐ Confirm that both raw and final processed data have been deposited in a public database such as GEO.

☐ Confirm that you have deposited or provided access to graph files (e.g. BED files) for the called peaks.

Data access links *May remain private before publication.*	*For "Initial submission" or "Revised version" documents, provide reviewer access links. For your "Final submission" document, provide a link to the deposited data.*
Files in database submission	*Provide a list of all files available in the database submission.*

Genome browser session (e.g. UCSC)	*Provide a link to an anonymized genome browser session for "Initial submission" and "Revised version" documents only, to enable peer review. Write "no longer applicable" for "Final submission" documents.*

Methodology

Replicates	*Describe the experimental replicates, specifying number, type and replicate agreement.*
Sequencing depth	*Describe the sequencing depth for each experiment, providing the total number of reads, uniquely mapped reads, length of reads and whether they were paired- or single-end.*
Antibodies	*Describe the antibodies used for the ChIP-seq experiments; as applicable, provide supplier name, catalog number, clone name, and lot number.*
Peak calling parameters	*Specify the command line program and parameters used for read mapping and peak calling, including the ChIP, control and index files used.*
Data quality	*Describe the methods used to ensure data quality in full detail, including how many peaks are at FDR 5% and above 5-fold enrichment.*
Software	*Describe the software used to collect and analyze the ChIP-seq data. For custom code that has been deposited into a community repository, provide accession details.*

Flow Cytometry

Plots

Confirm that:

☐ The axis labels state the marker and fluorochrome used (e.g. CD4-FITC).

☐ The axis scales are clearly visible. Include numbers along axes only for bottom left plot of group (a 'group' is an analysis of identical markers).

☐ All plots are contour plots with outliers or pseudocolor plots.

☐ A numerical value for number of cells or percentage (with statistics) is provided.

Methodology

Sample preparation	*Describe the sample preparation, detailing the biological source of the cells and any tissue processing steps used.*
Instrument	*Identify the instrument used for data collection, specifying make and model number.*
Software	*Describe the software used to collect and analyze the flow cytometry data. For custom code that has been deposited into a community repository, provide accession details.*
Cell population abundance	*Describe the abundance of the relevant cell populations within post-sort fractions, providing details on the purity of the samples and how it was determined.*
Gating strategy	*Describe the gating strategy used for all relevant experiments, specifying the preliminary FSC/SSC gates of the starting cell population, indicating where boundaries between "positive" and "negative" staining cell populations are defined.*

☐ Tick this box to confirm that a figure exemplifying the gating strategy is provided in the Supplementary Information.

Magnetic resonance imaging

Experimental design

Design type	*Indicate task or resting state; event-related or block design.*
Design specifications	*Specify the number of blocks, trials or experimental units per session and/or subject, and specify the length of each trial or block (if trials are blocked) and interval between trials.*
Behavioral performance measures	*State number and/or type of variables recorded (e.g. correct button press, response time) and what statistics were used to establish that the subjects were performing the task as expected (e.g. mean, range, and/or standard deviation across subjects).*

Acquisition

Imaging type(s) *Specify: functional, structural, diffusion, perfusion.*

Field strength *Specify in Tesla*

Sequence & imaging parameters *Specify the pulse sequence type (gradient echo, spin echo, etc.), imaging type (EPI, spiral, etc.), field of view, matrix size, slice thickness, orientation and TE/TR/flip angle.*

Area of acquisition *State whether a whole brain scan was used OR define the area of acquisition, describing how the region was determined.*

Diffusion MRI ☐ Used ☐ Not used

Preprocessing

Preprocessing software *Provide detail on software version and revision number and on specific parameters (model/functions, brain extraction, segmentation, smoothing kernel size, etc.).*

Normalization *If data were normalized/standardized, describe the approach(es): specify linear or non-linear and define image types used for transformation OR indicate that data were not normalized and explain rationale for lack of normalization.*

Normalization template *Describe the template used for normalization/transformation, specifying subject space or group standardized space (e.g. original Talairach, MNI305, ICBM152) OR indicate that the data were not normalized.*

Noise and artifact removal *Describe your procedure(s) for artifact and structured noise removal, specifying motion parameters, tissue signals and physiological signals (heart rate, respiration).*

Volume censoring *Define your software and/or method and criteria for volume censoring, and state the extent of such censoring.*

Statistical modeling & inference

Model type and settings *Specify type (mass univariate, multivariate, RSA, predictive, etc.) and describe essential details of the model at the first and second levels (e.g. fixed, random or mixed effects; drift or auto-correlation).*

Effect(s) tested *Define precise effect in terms of the task or stimulus conditions instead of psychological concepts and indicate whether ANOVA or factorial designs were used.*

Specify type of analysis: ☐ Whole brain ☐ ROI-based ☐ Both

Statistic type for inference (See Eklund et al. 2016) *Specify voxel-wise or cluster-wise and report all relevant parameters for cluster-wise methods.*

Correction *Describe the type of correction and how it is obtained for multiple comparisons (e.g. FWE, FDR, permutation or Monte Carlo).*

Models & analysis

n/a	Involved in the study
☐	☐ Functional and/or effective connectivity
☐	☐ Graph analysis
☐	☐ Multivariate modeling or predictive analysis

Functional and/or effective connectivity *Report the measures of dependence used and the model details (e.g. Pearson correlation, partial correlation, mutual information).*

Graph analysis *Report the dependent variable and connectivity measure, specifying weighted graph or binarized graph, subject- or group-level, and the global and/or node summaries used (e.g. clustering coefficient, efficiency, etc.).*

Multivariate modeling and predictive analysis *Specify independent variables, features extraction and dimension reduction, model, training and evaluation metrics.*