TECHNOLOGY AND CODE
published: 01 April 2022
doi: 10.3389/fcell.2022.842342

SuRVoS 2: Accelerating Annotation and Segmentation for Large Volumetric Bioimage Workflows Across Modalities and Scales

Avery Pennington[1], *Oliver N. F. King*[1], *Win Min Tun*[1], *Elaine M. L. Ho*[2], *Imanol Luengo*[1], *Michele C. Darrow*[2] and *Mark Basham*[1,2]*

[1]*Diamond Light Source Ltd., Didcot, United Kingdom,* [2]*The Rosalind Franklin Institute, Didcot, United Kingdom*

As sample preparation and imaging techniques have expanded and improved to include a variety of options for larger sized and numbers of samples, the bottleneck in volumetric imaging is now data analysis. Annotation and segmentation are both common, yet difficult, data analysis tasks which are required to bring meaning to the volumetric data. The SuRVoS application has been updated and redesigned to provide access to both manual and machine learning-based segmentation and annotation techniques, including support for crowd sourced data. Combining adjacent, similar voxels (supervoxels) provides a mechanism for speeding up segmentation both in the painting of annotation and by training a segmentation model on a small amount of annotation. The support for layers allows multiple datasets to be viewed and annotated together which, for example, enables the use of correlative data (e.g. crowd-sourced annotations or secondary imaging techniques) to guide segmentation. The ability to work with larger data on high-performance servers with GPUs has been added through a client-server architecture and the Pytorch-based image processing and segmentation server is flexible and extensible, and allows the implementation of deep learning-based segmentation modules. The client side has been built around Napari allowing integration of SuRVoS into an ecosystem for open-source image analysis while the server side has been built with cloud computing and extensibility through plugins in mind. Together these improvements to SuRVoS provide a platform for accelerating the annotation and segmentation of volumetric and correlative imaging data across modalities and scales.

Keywords: segmentation (image processing), annotation, U-net, volume electron microscopy (vEM), X-ray microscopy imaging, open source software, python (programming language), computer vision

OPEN ACCESS

Edited by:
Saskia Lippens,
Vlaams Instituut voor Biotechnologie, Belgium

Reviewed by:
Joris Roels,
Vlaams Instituut voor Biotechnologie, Belgium
Robert Haase,
TU Dresden, Germany

****Correspondence:***
Mark Basham
mark.basham@rfi.ac.uk

Specialty section:
This article was submitted to Cell Growth and Division, a section of the journal Frontiers in Cell and Developmental Biology

Received: *23 December 2021*
Accepted: *22 February 2022*
Published: *01 April 2022*

Citation:
Pennington A, King ONF, Tun WM, Ho EML, Luengo I, Darrow MC and Basham M (2022) SuRVoS 2: Accelerating Annotation and Segmentation for Large Volumetric Bioimage Workflows Across Modalities and Scales. Front. Cell Dev. Biol. 10:842342. doi: 10.3389/fcell.2022.842342

INTRODUCTION

The volume electron microscopy (vEM) and X-ray imaging ecosystems have flourished in recent years, through improvements to previously used techniques and the development of new techniques, all providing functional understandings through structural study (Peddie and Collinson 2014; Yoshiyuki et al., 2018; Xu et al., 2021). It is now common practice to collect 100s of GB of data daily across multiple correlative modalities contributing to the same project. This has shifted the experimental bottleneck to the image processing and analysis pipeline. Often, functional insights can only come from detailed segmentation and annotation of the image data, which currently is completed manually by an expert researcher. This creates a significant probability of creating "Dark Data," data that is collected, but due to time constraints and complexity, is not fully analyzed, or unintentionally analyzed in a biased way. Of specific concern are "Self Selection" and "Summaries of Data" (Hand, 2020), which are both routinely used to reduce the analytical load on the researcher. Our anecdotal experience supports this and shows that the vast majority of the data that is currently collected goes unanalyzed, or is only partially analyzed, and in many cases the analysis that is performed is qualitative or only takes into consideration a small number of examples or features due to the onerous nature of the task.

There are general computational tools available to ease some of the burden of the segmentation and image analysis process such as IMOD (Kremer et al., 1996), Ilastik (Berg et al., 2017), ImageJ (Abràmoff et al., 2004) and its many plugins, such as Weka (Hall et al., 2009), or commercial software such as Avizo (ThermoFisher.com, 2021). In addition, certain fields such as neuroanatomy have specialized image analysis and segmentation tools such as Knossos (Helmstaedter et al., 2011) that can perform some of these tasks. However, all these tools still often require specialized technical knowledge, substantial experimentation, and often scripting to adapt to complex data, especially data with artefacts or low signal-to-noise ratio. In all cases where machine learning is used, capturing a dataset of expert segmentation or annotations is a necessary first step to explore which methodologies provide the best results. This process works well if the team is composed of both biological and data science experts, however, there is still a significant challenge in providing sufficient amounts of high-quality annotation in a timely fashion.

Recently, several tools that allow non-specialists to use deep learning for image segmentation have been developed. CellProfiler 3.0 combines both a wizard-like interface for applying U-Net models and an extensive suite of tools for shallow machine learning-based segmentation and for analysis of the resulting segmentation (McQuin et al., 2018). CDeep3M is a cloud-based solution for applying deep learning to segmentation but has a command line interface (Haberl et al., 2018). DeepMIB provides a wizard-like interface for running a deep-learning pipeline but doesn't provide any support for developing annotation or analyzing the resulting segmentation (Belevich and Jokitalo, 2021). DeepImageJ, which, in conjunction with ImageJ and Fiji itself, provides a pipeline for prediction of U-Net-based segmentation models as well as many tools for the preparation of training data and the analysis of results (Gomez-de-Mariscal et al., 2021). SuRVoS2 is unique in providing a self-contained GUI tool designed for non-programmers that allows painting of annotations using super-regions, a machine-learning based prediction mechanism for accelerating the annotation process, an integrated system for training and prediction of a U-net model, and tools for analyzing the resulting segmentation.

SuRVoS Workbench (Darrow et al., 2017; Luengo et al., 2017) was originally developed to address this need for an accelerated process of producing initial expert segmentation on which to base subsequent machine learning methods. Annotation in SuRVoS is based around the concept of supervoxels (Lucchi, et al., 2011) which provide a way to select a large number of voxels in 3D with little user effort, yet still respect the boundaries found within the data itself. After using supervoxels to quickly annotate regions within a volume, an iterative shallow machine learning strategy with integrated filters for data augmentation could be applied to predict the label assignments of the volume (**Figure 1**). This involves pre-calculating image features and training a machine learning method such as random forests or SVM (Support Vector Machine) on the provided scribble annotation, and then to predict the segmentation of the whole volume. Because the image features are calculated separately from the machine learning algorithm, this method is considered a "shallow" machine learning approach. A novel hierarchical strategy could restrict annotations and predictions based on parent-child relationships and a "label-splitter" functionality could be used to separate out objects based on their inherent properties for analysis.

Overall, SuRVoS provided a processing pipeline which sped up the segmentation process, in some cases enabling research which wouldn't have been possible if segmentation was completed manually (Strotton et al., 2018). However, with increasing data sizes and rates of collection, SuRVoS became cumbersome due to local memory requirements. We also received multiple requests for usability improvements, especially around installation on various operating systems. And finally, our research and implementation of deep machine learning strategies has advanced. Together, these reasons have motivated the release of SuRVoS2 as both an API backend server and a new client based on the Napari (Napari Contributors, 2021) ecosystem. This edition includes a new client-server architecture, integration of Dask (Rocklin, 2015) for parallelized image processing, and both semi-automatic shallow and deep machine learning pipelines.

METHODS

Segmentation of large 3D volumetric imaging data presents challenges in providing computing resources, project management, and in visualization and interactivity. SuRVoS2 (licensed under Apache 2.0; Basham, 2021) approaches these challenges with the ethos of providing open-source, free-to-use options for accelerated annotation, segmentation and analysis that are data type agnostic.

Client-Server Architecture

Volumetric image segmentation is computationally intensive, but by way of its implementation as a client-server application, SuRVoS2 can be run on high-performance servers (e.g. multi-processor, multi-GPU, or high random access memory (RAM) machines) yet provide an interactive experience for users. The server can be used to run compute-heavy operations, such as segmentation algorithms, while the client is run on a convenient device local to the user, such as a laptop. The client connects to the server and directs the segmentation workflow, and all the intermediate files are stored on a server-accessible file system, with the client downloading only the data needed for interactive segmentation and visualization.

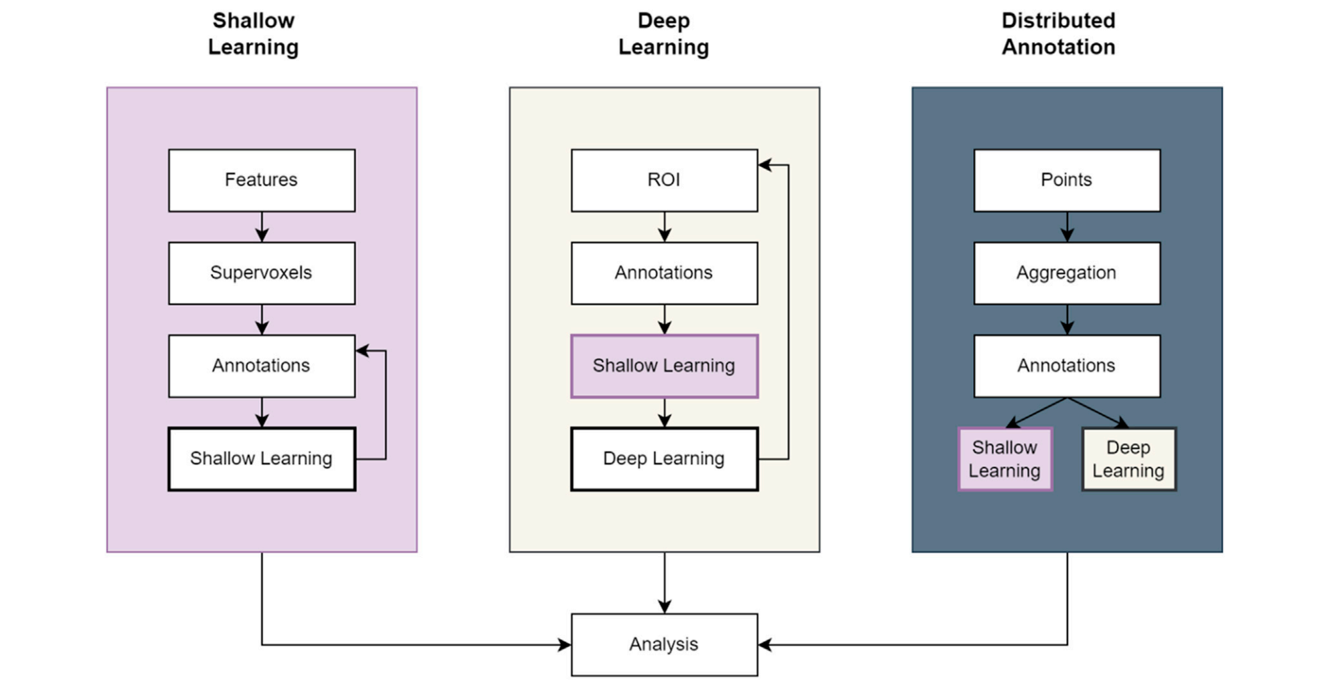


FIGURE 1 | Example workflow pipelines in SuRVoS2. The Shallow Learning pipeline is the same as previously available in SuRVoS using an iterative painting and predicting cycle to produce an output segmentation. The Deep Learning pipeline incorporates the Shallow Learning pipeline to quickly generate expert segmentations on a region of interest (ROI) which can subsequently be used to train a deep learning model for application to the full volume. And finally, the Distributed Annotation pipeline uses geometric data alongside either the Shallow Learning or Deep Learning pipelines to segment objects marked by 3D points.

The client uses a REST (Representational State Transfer) API implemented with the Hug Library (Crosley, 2016) to make requests to the SuRVoS2 server. The REST API uses HTTP (Hypertext Transfer Protocol) to implement a clean interface for functionality such as generating features, segmentation, and analysis. SuRVoS2 can be run headless on a cluster or virtual machine and then the Napari-based client connects over HTTP. This allows the server to run as a service on HPC-based clusters, for example, and could be containerized using Docker (Merkel, 2014) in order to deploy it at large scale in one of the major cloud providers. This enables the core functionality of SuRVoS2 to be run on appropriate computational resources with fast access to the image data resulting in a faster process for the user and enabling the processing of much larger datasets, limited only by server RAM. Additionally, the integration of the REST API and separating the core logic from the user interface allows for future extension of the client side to other devices or browser-based clients, which would further increase access to SuRVoS2 tools and functionality.

Workspace Philosophy

The SuRVoS2 server gathers and manages the information required throughout a complex segmentation project through the use of "Workspaces." A workspace is created from an imaging volume and as the segmentation workflow of choice is used (**Figure 1**), all objects created (features, annotations, supervoxels, segmentation pipelines and analysis outputs; *See* **Table 1**) and the state of the workflow are stored in the workspace file system. The parameters used throughout the workflow can be saved and loaded into a new workspace for application to a naïve dataset. The data objects are generally stored as chunked HDF5 (Hierarchical Data Format 5) files and manipulated using the h5py library (Colette et al., 2017). This chunking mechanism can be globally tuned, allowing the file access performance to be optimized for a given set of hardware.

Napari as a Graphical User Interface (GUI)

Napari is "a fast multi-dimensional image viewer designed for browsing, annotating and analyzing large multi-dimensional images" (Napari Contributors, 2021). It is a well-implemented PyQT-based application framework with a broad user community, which provides an extensible base for building application-specific annotation and segmentation tools. Napari is encouraging the growth of an ecosystem of additional image processing plugins providing features such as animation (Sofroniew, 2021), tracking (Prigent, 2021), and deconvolution (Perdigao, 2021). By utilizing Napari as a common GUI, a user can access a range of tools from multiple authors but following similar conventions for improved ease-of-use and inter-operability.

A key benefit of an interactive GUI for performing image processing operations is the ability to interactively explore an image dataset. Napari provides a broad set of tools for 2D and 3D inspection of the image data, interactive painting of annotations, and visualization of outputs from each stage of a processing pipeline. Additionally, Napari supports overlaying multiple floating point image layers, each with controllable opacity,

TABLE 1 | List of all filters/features, shallow and deep learning, and label splitter/analysis options within SuRVoS2.

Filters and features	Machine learning	Label splitter/Analysis
Produce output images for use in shallow or deep learning pipelines	Uses manual or shallow learning created training annotations to predict classes for other voxels	Inherent characteristics of data or segmented objects used to separate objects into groups
Basic Features:	Shallow Learning:	Mean Intensity
Simple Invert	Random Forest	Standard Deviation of Intensity
Invert Threshold	Extra Random Forest	Variation of Intensity
Threshold	Gradient Boosting	Volume
Rescale	Support Vector Machine (SVM)	Bounding Box Volume
Gamma Correct	Active Contour without Edges (ACWE)	Log Bounding Box Volume
Blob:	Watershed	Position X
Structure Tensor Determinant	Deep Learning:	Position Y
Frangi	2D U-Net	Position Z
Hessian Eigenvalues	3D U-Net [a]	Bounding Box Depth
Denoising:	FPN [a]	Bounding Box Height
Total Variation Denoise	—	Bounding Box Width
Gaussian Blur	—	Oriented Bounding Box Volume
Median	—	Log Oriented Bounding Box Volume
Wavelet	—	Oriented Bounding Box Depth
Edges:	—	Oriented Bounding Box Height
Spatial Gradient 3D	—	Oriented Bounding Box Width
Difference of Gaussians	—	—
Laplacian	—	—
Morphology:	—	—
Dilation	—	—
Erosion	—	—
Closing	—	—
Euclidean Distance Transform	—	—
Skeletonize	—	—
Neighborhood:	—	—
Gaussian Norm	—	—
Gaussian Centre	—	—

[a]*Indicates options where training is currently done using the SuRVoS2 API externally to the graphical user interface (GUI), with the deep learning module for prediction available within the SuRVoS2 GUI.*

which can be helpful when viewing two or more registered images together. In addition, integer-valued annotation layers and geometry layers can be viewed together with floating point image layers allowing many different data types to be displayed concurrently for correlational painting of segmentations and labelling of objects. Using the 3D mode provides options for 3D rendering of data and segmented objects including isosurfaces and attenuated maximum intensity projection.

Napari and the SuRVoS2 plugin can run on Linux, Windows, and Macintosh operating systems (*see* https://github.com/DiamondLightSource/SuRVoS2 for code, installation instructions and documentation). This arrangement of SuRVoS2 as a Napari plugin both extends SuRVoS2 with access to 2D and 3D visualization and annotation tools that Napari offers and likewise, extends Napari by providing a complete system for managing segmentation and image analysis workflows, including an extensive set of filters, supervoxels, and shallow and deep machine learning segmentation options.

Machine Learning Implementations

SuRVoS2 provides both a shallow and a deep machine learning pipeline (**Figure 1**). The shallow learning pipeline is equivalent to the functionality previously available in SuRVoS (Darrow et al., 2017; Luengo et al., 2017) whereby an initial set of training annotations are created and then an image segmentation model consisting of supervoxel-based image features and a random forest or SVM model is trained and evaluated, and then additional annotations or refinements of annotations are produced, and the model retrained and re-evaluated. The deep learning pipeline uses the 2D U-net from FastAI (Howard and Gugger, 2020) as a base and implements this functionality using The Kornia library (Riba et al., 2020) which was built using the Pytorch library (Paszke, et al., 2019), the Torch-IO library, some machine learning functions from scikit-learn (Pedregosa et al., 2011) and some image processing operations in SuRVoS2 also use the Scipy ndimage package (Virtanen et al., 2020) and the scikit-image library (van der Walt et al., 2014). These image processing technologies allow for GPU-accelerated computing to accommodate large datasets. Some additional image processing operations in SuRVoS2 use the Scipy ndimage package (Virtanen et al., 2020). The deep learning pipeline available in SuRVoS2 uses a region of interest (ROI) system whereby smaller, more manageable volumes are segmented either manually or through the shallow learning pipeline and used as training data. By providing one or more segmented ROIs as training data, the deep learning pipeline can transfer this learning to the rest of the volume. A multi-axis 2D U-Net (King, 2021) is used to predict the output segmentation in different directions, combining the output into a final segmentation. To enable faster processing, Dask is used for parallelized image processing of chunked data. Together these features of SuRVoS2 enable the efficient processing of large datasets.

SuRVoS2 API

Complex segmentation problems usually require custom workflows. A Python API for SuRVoS2 can be accessed from either Jupyter notebooks (Kluyver et al., 2016), Python scripts or any other client capable of performing HTTP requests. A set of Jupyter-specific convenience functions allows for the inspection, creation, and modification of data within a SuRVoS2 workspace and give control of SuRVoS2 segmentation pipelines (*see* https://github.com/DiamondLightSource/SuRVoS2 for a testing notebook with examples). The output of these pipelines can be loaded into a common workspace for visualization and evaluation in the SuRVoS2 GUI. Additionally, SuRVoS2 is extensible by end users through a plugin mechanism composed of two parts: an API and a GUI. The API is implemented as a Python server using the Hug library (Crosley, 2016) to provide the REST API allowing for access to the workspace and all data in it. The GUI is a separate Python file consisting of the PyQT-based (RiverbankComputing.com, 2021) widgets that will be rendered in a tab within the SuRVoS2 user interface. For example, the SuRVoS2 ROI plugin has an API module that stores the ROI on the server and that can create and delete ROIs. Then it has a GUI module that contacts the API, gets the current list of ROIs, displays it, and allows the user to interactively create and delete ROIs.

RESULTS

SuRVoS2 implements multiple new features which accelerate image annotation, segmentation, and analysis. Example case studies have been sourced to highlight these features in the context of vEM and correlative imaging techniques. First, X-ray microCT of human placenta will be used to highlight the deep learning pipeline (Tun et al., 2021); second, cryo soft X-ray tomography (cryoSXT) of *Trypanosoma brucei* will be used to demonstrate the label splitting and analysis functionalities; third, cryo electron tomography of a virus-infected cell will be used to illustrate the display and manipulation of non-mask based data such as the output from a distributed citizen science-based annotation workflow; fourth, correlative cryoSXT and cryo structured illumination microscopy datasets of a virus-infected cell will be used to demonstrate the ways correlative datasets can be used in the SuRVoS2 pipeline; and finally, an example Jupyter notebook will be used to highlight an advanced implementation of the SuRVoS2 API for clustering, segmentation, and visualization within the GUI.

Deep Learning With the SuRVoS2 Implementation of a 2D U-Net

The two main segmentation methods available in SuRVoS2 can be used together to rapidly generate high quality segmentations for large datasets (Alvarez-Borges et al., 2021; Tun et al., 2021). SuRVoS2 implements a version of the concept of “weak annotation” in which scribble-based annotation is used to train a shallow machine learning model and then the prediction of that model is used as ground truth to train a deep learning U-Net (Lin D. et al., 2016; Khoreva et al., 2017; Li et al., 2018). This approach requires expert evaluation of the output predictions of both the shallow and deep machine learning models and is classified as a semi-automatic method. A subvolume (ROI) of the dataset is first selected, then supervoxels are generated to aid fast annotation of this ROI. This subvolume annotation can be done entirely manually, utilizing the benefits of supervoxels, or partial annotations, in conjunction with extracted image features, can be used to train an ensemble (e.g. a random forest) or an SVM classifier. The trained classifier is then used to predict the missing annotations within the ROI. After this step, the resulting segmented ROI and the corresponding data can then be used as “expert segmentation” data to train a U-Net network. This trained U-Net model is then available to use for predicting segmentation of the entire large dataset (*See* Tun et al., 2021 for validation of this methodology).

U-Net models have advantages over ensemble or SVM methods (Ronneberger et al., 2015; Seo et al., 2020) in terms of generalizing better to new data and in not requiring user-led extraction of image features for training and prediction. The U-Net, by contrast, learns the features to extract from the data during model training. However, deep learning models like the U-Net often require much larger amounts of training data to perform well. Therefore, the practical application of deep learning for segmentation suggests the use of a shallow machine learning pipeline, trained quickly using supervoxels, for the creation of training data that can be used as input to the deep learning pipeline.

To demonstrate the above pipeline, an X-ray micro-tomography dataset (EMPIAR 10562, Tun et al., 2021) collected from a 3 × 3 × 3 mm sample of human placental tissue at Diamond Light Source beamline i13-2 was used (for more details of sample preparation and data collection *see* Tun et al., 2021). The full dataset has dimensions 2520 × 2520 × 2120 pixels and a 256 × 256 × 256 pixel ROI was selected from this dataset using the SuRVoS2 data previewer (**Figure 2A**). A workspace was created from the ROI and the data was denoised using a total variation filter (**Figure 2B**) before supervoxels with an average shape of 10 × 10 × 10 pixels were generated from this denoised volume (**Figure 2C**). A paintbrush tool was used to annotate some regions in the data (**Figure 2D**), separating the background supervoxels (shown in red) and those representing blood vessels (shown in blue). Further feature images were created from the ROI, namely a Gaussian blur, Hessian eigenvalues and normalized Gaussian datasets (not shown). The supervoxel annotations were used, along with the extracted image feature datasets, to train a random forest classifier which was then used to predict the labels for all of the supervoxels in the ROI (**Figure 2E**). At this point, the segmentation of the 256 × 256 × 256 pixel ROI is broadly correct with some minor misclassifications. This ROI segmentation was used to train a 2D U-Net model using an approach which leverages the 3D nature of

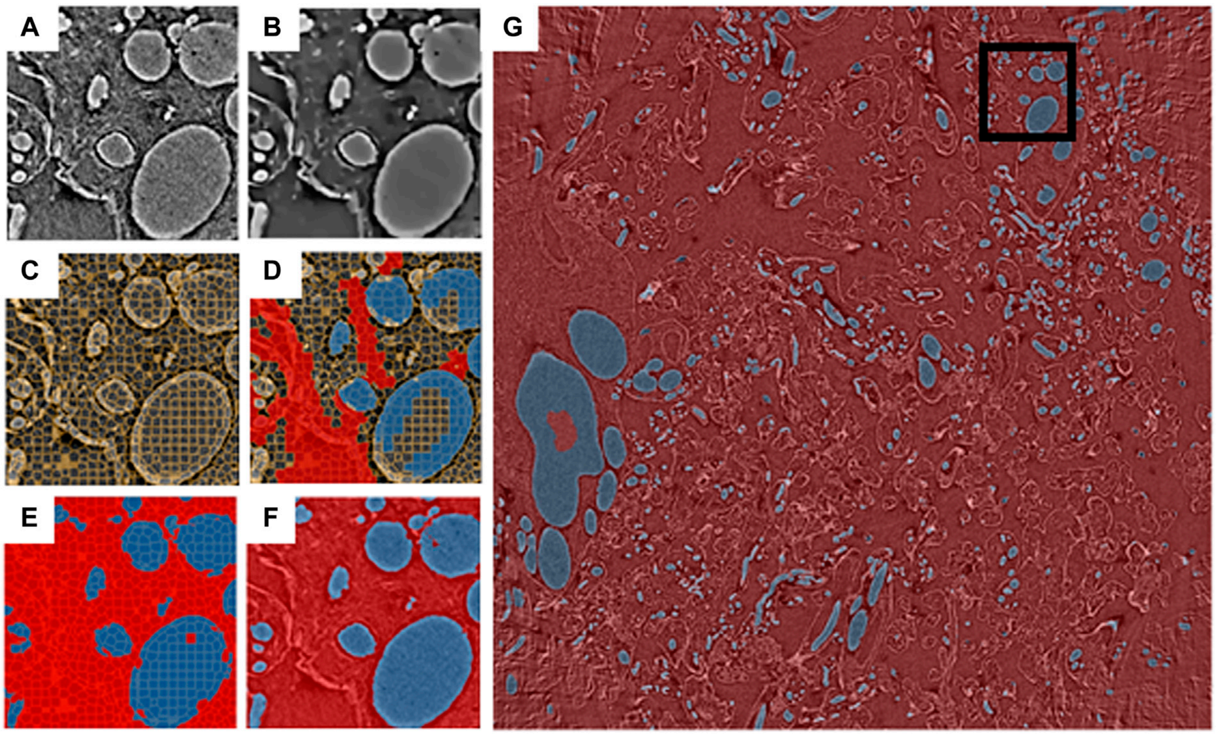


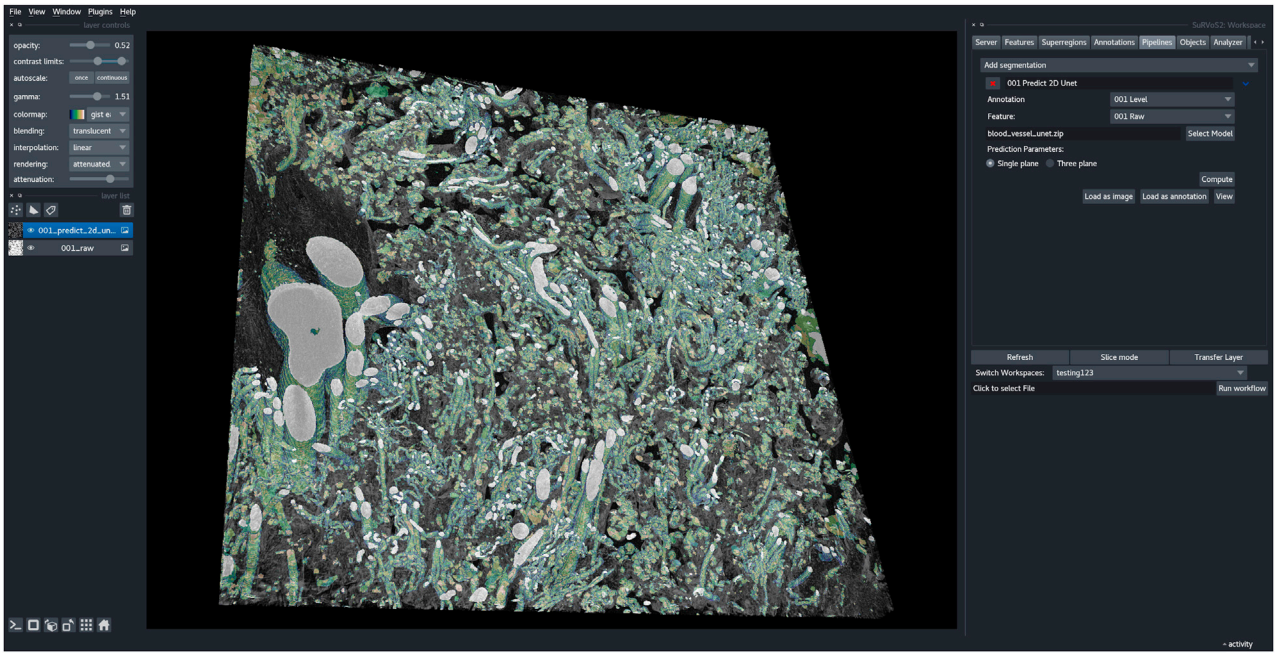
FIGURE 2 | Blood vessels from an X-ray micro-tomography dataset of human placenta segmented using the SuRVoS2 deep learning pipeline. Images **(A–F)** show a central slice of a 256 × 256 × 256 pixel region of interest (ROI), marked in **(G)** with a black box. **(A)** Raw data. **(B)** Data with a total variation denoising filter applied. **(C)** Supervoxels generated from the denoised data. **(D)** Annotations applied to the supervoxels (red: background, blue: blood vessels). **(E)** Supervoxel-based prediction of segmentation labels using a random forest classifier trained on the annotations shown in **(D)**. **(F)** Voxel-based prediction of segmentation labels using a 2D U-Net model trained on the segmentation output from **(E)** and the raw data **(A)**. **(G)** Voxel-based predictions of segmentation labels for a central 100 slices of the full 2520 × 2520 dataset using the 2D U-Net model trained as described in **(F)**.

the data (Alvarez-Borges et al., 2021). To do this, 2D images from the raw data and corresponding segmentation label volumes were sliced in three orthogonal planes to create the stack of training images. This process yielded 768 images of 256 × 256 pixels with the corresponding "ground truths." These are enriched further by augmentations such as flips, contrast adjustments, and geometric distortions. Once trained, the process creates an output by taking the volume to be predicted and again dividing it into three stacks of 2D images, and augmenting this with rotations. This results in each voxel being classified multiple separate times and a voting system is used to select the final outcome of the voxel segmentation. The predicted segmentation of the ROI generated by this U-Net model has improved on the shallow learning segmentation output, addressing many of the misclassifications found there (**Figure 2F**).

Next, a new workspace was created containing a central portion of the full-size dataset (2520 × 2520 × 100 pixels). The U-Net model, trained as described above, was then used to generate segmentations from this larger volume of data (**Figure 2G**). Utilizing the capabilities of the Napari viewer and SuRVoS2 plugin, these segmentations can be rendered in 3D and overlaid with the image data to clearly visualize the now segmented blood vessels (**Figure 3**). Due to the large data size, manual segmentation of this single volume is estimated to have taken approximately 320 h, or around 2 months of person-time. Using the initial, shallow learning implementation of SuRVoS was estimated to have reduced the time spent by half, still taking approximately 1 month of person-time. Using the deep learning pipeline in SuRVoS2, segmentation of this volume was reduced to approximately 1.2 h of person-time to segment the ROI using the shallow learning pipeline and 4 h of computational time on a high performance machine consisting of two Intel® Xeon® Gold 6242R processors each with 20 cores running at 3.1 GHz, and 768 GiB of system memory. The GPU used was an NVIDIA Tesla V100 with 32 GB of available memory.

Data Analysis in SuRVoS2

During segmentation, the class each object is assigned to is often left to the individual researcher based on 2D examination of the objects present in the data, leading to potential subjectivity in the results. The Label Splitter tool was developed as a means to derive classifications directly from the data using the inherent characteristics of each object, such as size, shape, intensity, variation, etc. Category labels and rules to generate classes are still decided by a user, meaning some bias or subjectivity is still present, however it will now be consistently applied, even across multiple normalized datasets.



opacity: 0.52
contrast limits:
autoscale: once continuous
gamma: 1.51
colormap: gist e
blending: translucent
interpolation: linear
rendering: attenuated
attenuation:

001_predict_2d_un...
001_raw

001 Predict 2D Unet
Annotation: 001 Level
Feature: 001 Raw
blood_vessel_unet.zip
Prediction Parameters:
[✓ Single plane] Three plane
Switch Workspaces: testing123

FIGURE 3 | SuRVoS2 plugin integrated into the Napari viewer. The output blood vessel segmentations from the deep learning pipeline in SuRVoS2 visualized using the 3D rendering option and overlaid on the raw data in the Napari viewer.

To demonstrate the functionality of the Label Splitter tool, a cryoSXT dataset of *Trypanosoma brucei* collected from Diamond Light Source beamline B24 was used (*see* Darrow et al., JoVE, 2018 for more information). Starting from previously segmented objects, the Label Splitter tool displays information about each object that can be used to define rules for separating the objects into classes (**Figure 4** and **Table 1**). These rules can be interactively developed using the table and graph view. The graph is split by a line at the location of the chosen rule value and double clicking on an entry in the table will take the user to the object in the data allowing for clear, specific delineations of classes of object based on the inherent characteristics of the objects.

Rules are applied sequentially to all selected objects creating bespoke classes of objects. After splitting, each new class can be visualized and given a unique label and color (**Figure 5**). The Label Splitter tool can be used on annotation layers regardless of their provenance (manual, shallow learning, deep learning or geometric) and it can also be used iteratively using the output of another Label Splitter instance to create complex and nested hierarchies that can accommodate data with large numbers of different types of objects.

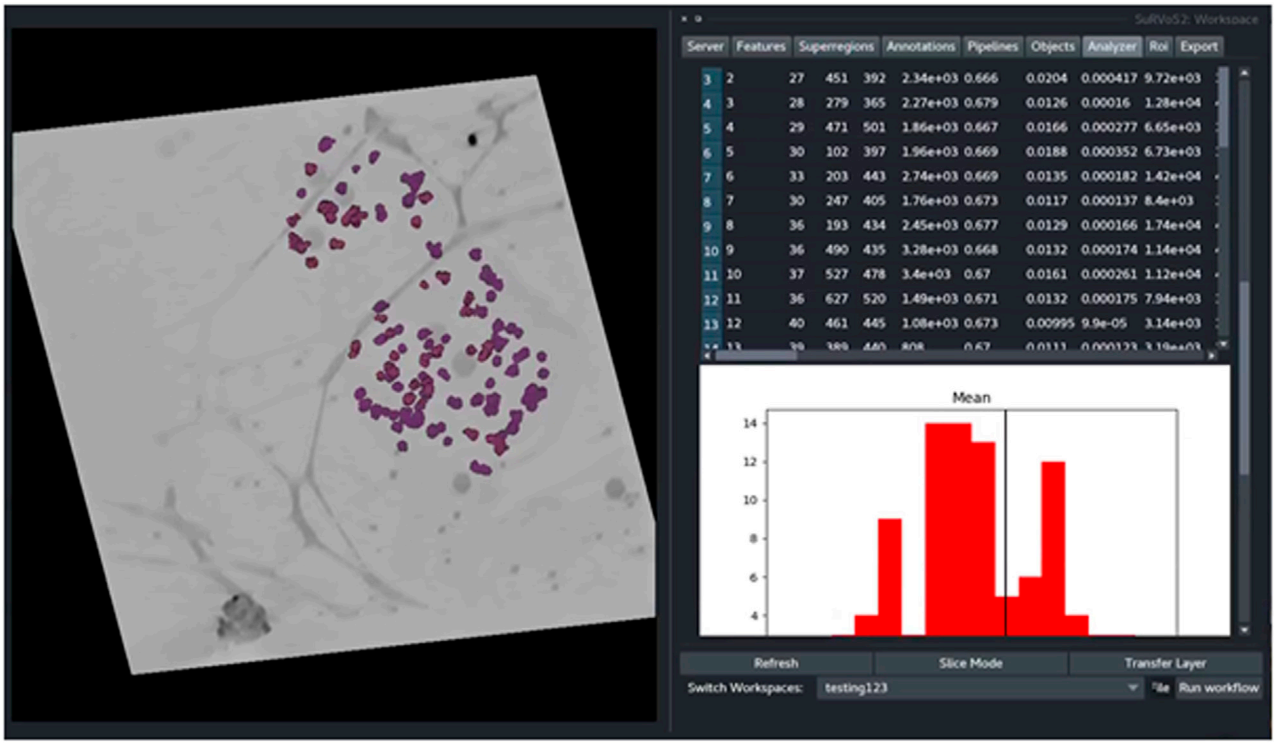
Geometric Data in Segmentation Workflows

SuRVoS2 supports segmentation projects that utilize geometric point data which can be visualized, edited, and used to generate mask-based annotations. Geometric data can be generated in many different ways, for example point-based locations of particles of interest in a 3D volume produced manually or automatically for sub-tomogram averaging (Wagner et al., 2019; Bepler et al., 2020); or through object detection workflows specific to an organelle or other object (Wei et al., 2020), or even points placed on images as part of distributed annotation workflows within a lab or utilizing citizen science platforms (Zooniverse, 2019).

Geometric data is an efficient way to encode knowledge about an image, where a point indicates both an object's class membership and its location. To demonstrate the use of SuRVoS2 with geometric data, point-based data gathered from a crowdsourced workflow focused on finding viruses inside of a cell using cryo electron tomography (cryoET) data collected at Diamond Light Source on eBIC was used (Sutton et al., 2020)

It is possible to either create or import geometric data using Napari. In this case, the data were imported using a simple CSV file format. Points can then be viewed (2D and 3D) and edited/deleted (2D only) while overlaid with the raw image data as reference using the SuRVoS2 plugin (**Figure 6**). Point data can have an encoded class label represented by the point color. The table view lists the coordinates of the 3D point and its class. Double-clicking on the row of a particular point translates the current view to that location, centering the point (**Figure 6**). Together these capabilities provide users the tools needed to evaluate and refine crowdsourced annotations and those generated by automated object detection workflows (Jaeger, 2018).

The Rasterize Points plugin in SuRVoS2 can convert point-based data into segmentation masks by painting ellipsoidal blobs on the 3D point locations. The scale and orientation of the ellipsoids can be set and this operation allows the ellipsoid to be initialized as an Active Contour Without Edges (Chan and



3	2	27	451	392	2.34e+03	0.666	0.0204	0.000417	9.72e+03
4	3	28	279	365	2.27e+03	0.679	0.0126	0.00016	1.28e+04
5	4	29	471	501	1.86e+03	0.667	0.0166	0.000277	6.65e+03
6	5	30	102	397	1.96e+03	0.669	0.0188	0.000352	6.73e+03
7	6	33	203	443	2.74e+03	0.669	0.0135	0.000182	1.42e+04
8	7	30	247	405	1.76e+03	0.673	0.0117	0.000137	8.4e+03
9	8	36	193	434	2.45e+03	0.677	0.0129	0.000166	1.74e+04
10	9	36	490	435	3.28e+03	0.668	0.0132	0.000174	1.14e+04
11	10	37	527	478	3.4e+03	0.67	0.0161	0.000261	1.12e+04
12	11	36	627	520	1.49e+03	0.671	0.0132	0.000175	7.94e+03
13	12	40	461	445	1.08e+03	0.673	0.00995	9.9e-05	3.14e+03
14	13	39	389	440	808	0.67	0.0111	0.000123	3.19e+03

FIGURE 4 | The Label Splitter tool for application of classification rules using inherent characteristics of the segmented objects. Information about each object is displayed on a table and graph. The graph is split by a line at the location of the chosen rule value and double clicking on an entry in the table takes the user to the chosen object in the data.

Vese, 2001) object which can expand or contract, fitting the image data. This or more complex conversion strategies for computationally converting point-based data into segmentations can be designed using the machine learning pipelines within SuRVoS2 (**Figure 1**).

Extended Functionality Using the SuRVoS2 API

SuRVoS2 includes a Python API that can be used from Jupyter Notebooks, allowing the notebook to access, process, and add information into SuRVoS2 workspaces through the SuRVoS2 server (**Figure 7**). Additional functionality implemented through the API includes clustering and visualization of patches sampled from the volume. For example, a set of geometric points is used as sampling locations for uniformly sized patches (e.g. 64 × 64 × 64 pixels) and image features can be computed for each patch using a 2D ResNet model (He, et al., 2015). The features can then be clustered and an embedding made to visualize the features on a 2D plot using Unified Manifold Approximation and Prediction (UMAP; McInnes, et al., 2020) or TSNE (Van der Maaten and Hinton, 2008) (**Figure 7**).

As a second example also starting from geometric data with location and class information, the SuRVoS2 API can be used from any Jupyter client to segment the objects using additional deep learning strategies (such as a 3D U-Net or Feature Pyramid Network (FPN; Lin T et al., 2016) **Figure 7**). The training of the model is performed through the Jupyter notebook, a prediction is made, and the results are visualized as a layer in the SuRVoS2 GUI to allow for inspection of the segmentation. The ability to call standard and additional SuRVoS2 functionality external to the GUI extends the addressable use-cases and provides any researcher with Python programming skills the ability to create bespoke, specialized processing pipelines to address their specific annotation, segmentation, and analysis needs.

Multimodal Correlation of 3D Datasets in SuRVoS2

SuRVoS2 supports the use of correlative, multi-modal imaging data for visualization and annotation. Multiple datasets can be loaded into a workspace and viewed in 2D or 3D as individual layers. The datasets must already be appropriately transformed as this functionality is not currently available in either Napari or SuRVoS2. To demonstrate the use of SuRVoS2 with correlative, multi-modal data, cryoSXT and cryo structured illumination microscopy (cryoSIM) of virus infected cells were aligned externally and displayed within SuRVoS2 (**Figure 8**; EMPIAR 10416 and S-BIAD19 respectively). The opacity of each layer can be controlled allowing for display of both 3D layers simultaneously.

This functionality allows the user to reference one dataset when annotating another dataset, for example using the

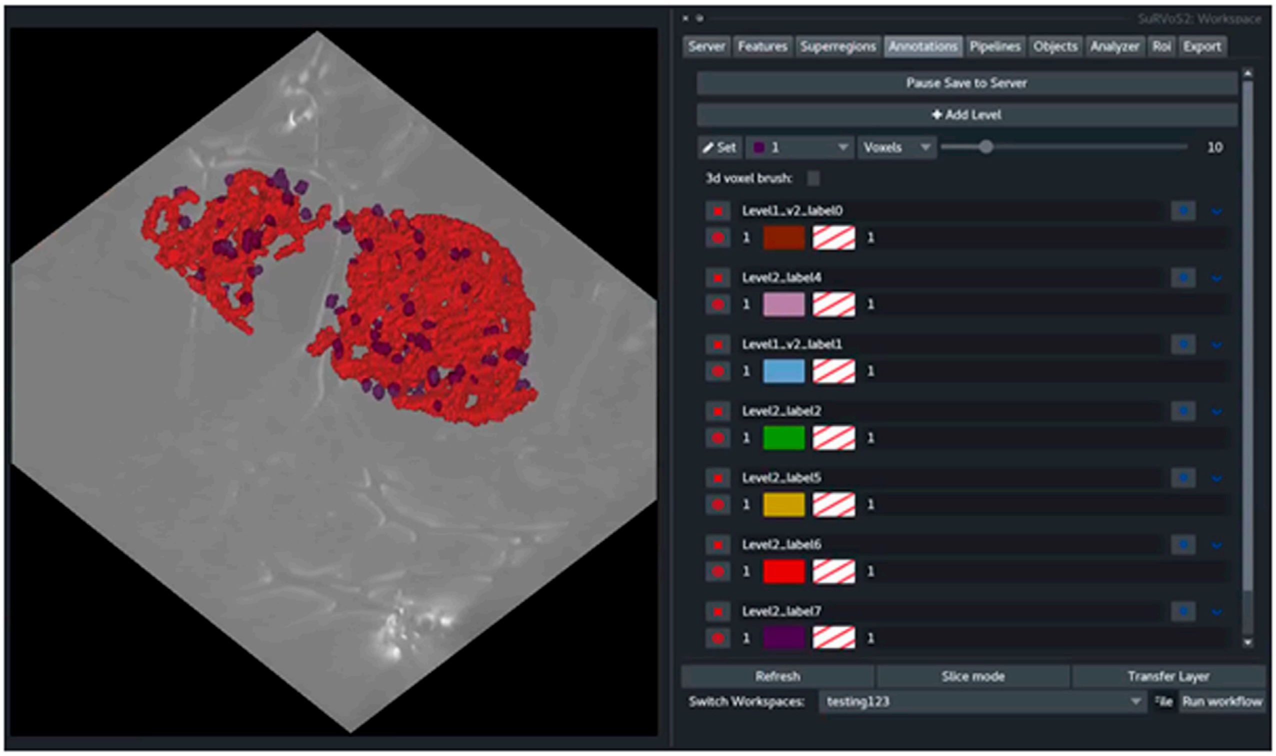
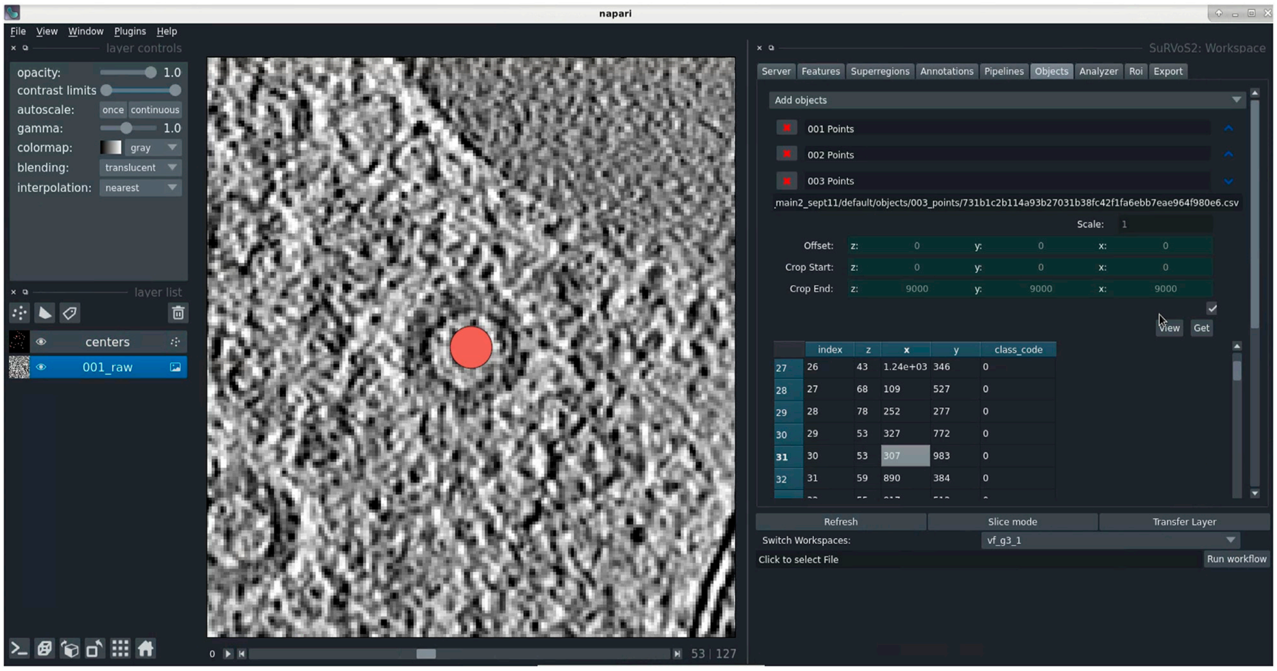


FIGURE 5 | The output of the Label Splitter can be visualized as annotation layers within the SuRVoS2 plugin. Rules are applied sequentially to all selected objects and after splitting, each new class can be visualized and given a unique label and color.



opacity: 1.0
contrast limits
autoscale: once continuous
gamma: 1.0
colormap: gray
blending: translucent
interpolation: nearest

main2_sept11/default/objects/003_points/731b1c2b114a93b27031b38fc42f1fa6ebb7eae964f980e6.csv

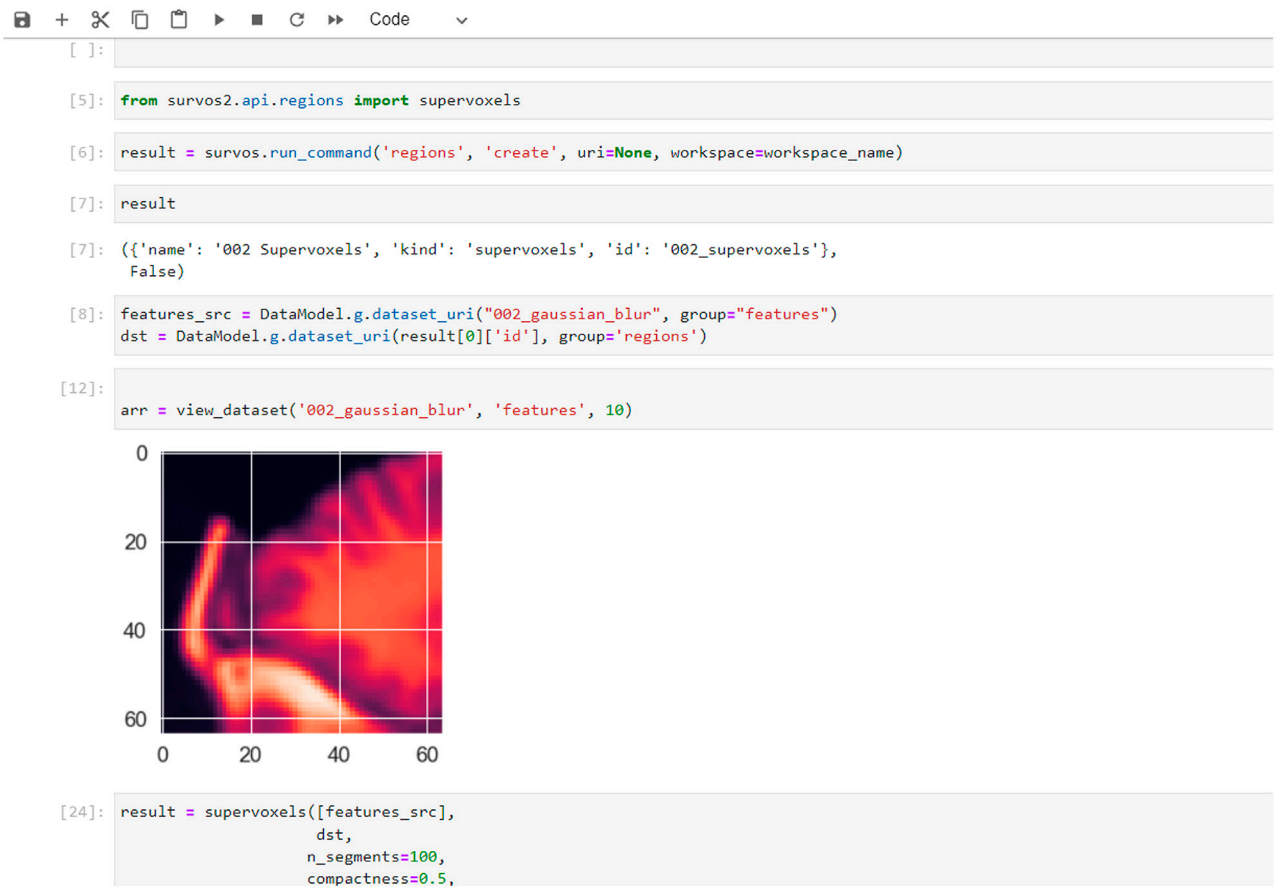
Scale: 1

Offset: z: 0 y: 0 x: 0
Crop Start: z: 0 y: 0 x: 0
Crop End: z: 9000 y: 9000 x: 9000

	index	z	x	y	class_code
27	26	43	1.24e+03	346	0
28	27	68	109	527	0
29	28	78	252	277	0
30	29	53	327	772	0
31	30	53	307	983	0
32	31	59	890	384	0

Switch Workspaces: vf_g3_1

FIGURE 6 | Use of geometric data in SuRVoS2 to analyze crowdsourced annotations. **(A)** Data were imported into the Napari viewer using a simple CSV file format indicating 3D centroid locations and object class. The tools in SuRVoS2 can be used to visualize, edit, and delete geometric data, including a function to take the user directly to an object of interest by double clicking on its entry in the table view.



```
[ ]:

[5]: from survos2.api.regions import supervoxels

[6]: result = survos.run_command('regions', 'create', uri=None, workspace=workspace_name)

[7]: result

[7]: ({'name': '002 Supervoxels', 'kind': 'supervoxels', 'id': '002_supervoxels'},
      False)

[8]: features_src = DataModel.g.dataset_uri("002_gaussian_blur", group="features")
     dst = DataModel.g.dataset_uri(result[0]['id'], group='regions')

[12]:
     arr = view_dataset('002_gaussian_blur', 'features', 10)
```

```
[24]: result = supervoxels([features_src],
                           dst,
                          n_segments=100,
                          compactness=0.5,
```

FIGURE 7 | Examples of additional features and use-cases available when using the SuRVoS2 API from within Jupyter notebooks. Example of the SuRVoS2 API being used through a notebook with results which can be visualized using the SuRVoS2 GUI.

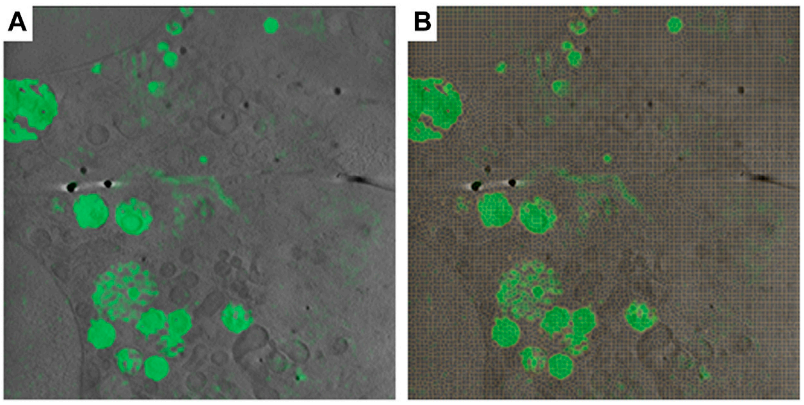


FIGURE 8 | 3D CryoSXT with correlated 3D cryoSIM in SuRVoS2. **(A)** Multimodal imaging can be displayed together as individual layers with control over opacity and color. **(B)** SuRVoS2 specific functionality, such as supervoxels (shown in yellow) can be calculated using one of the datasets for use during segmentation or annotation of the other.

fluorescence to guide segmentation of specific organelles. It is also possible to use other Napari and/or SuRVoS2 functionality to accelerate the annotation or segmentation process. For example, a deconvolution algorithm could be applied (e.g. RedLionfish (Perdigao, 2021) in Napari) to the fluorescence data, followed by calculation of supervoxels and then segmentation of areas of fluorescence in the non-fluorescence dataset. It is also possible to use the 3D fluorescence dataset as an input feature map during the shallow or deep learning pipelines or as a path to generation of point-based geometric data, although more research is needed to understand best-practices for these applications.

DISCUSSION

Segmentation is a difficult and time-consuming task that absorbs the attention and expertise of domain experts and software engineers alike. The general principles which have led to successful SuRVoS2 segmentations include using step-wise and hierarchical approaches. Rather than attempting to segment all objects in the field of view at once, a more successful strategy may be to segment large regions or distinctive objects from the background as a first step. This can then be followed by additional iterations through the segmentation pipelines, now utilizing the parent-child hierarchy to limit annotation and segmentation calculations to only regions where the objects of interest are to be found. By breaking the problem up into smaller chunks, the semi-automated computational strategies are generally more successful. It is also helpful during the expert annotation stage to be mindful of class imbalances. If an object of interest is only present a small number of times, or only represented by a small number of voxels in the training data, it will be difficult for the deep learning pipeline to accurately detect these objects in the remaining volume. Areas for expert annotation should be chosen carefully to ensure they represent the remaining volume, and secondary areas may be required to emphasize objects which do not occur often.

SuRVoS2 has several key strengths and provides a unique combination of tools that support this iterative process required for producing accurate output segmentations. First, SuRVoS2 encourages interactive visualization and exploration of image volumes and associated geometric data in both 2D and 3D within the user-friendly Napari environment. This is useful when understanding what objects are present in the data and choosing representative ROIs. Second, SuRVoS2 provides several different mechanisms to paint and prepare annotations, including supervoxels for accelerated annotation. And finally, SuRVoS2 provides both a fast mechanism for sparsely training a shallow model to predict annotations as well as a state-of-the-art deep U-Net segmentation model that can learn on densely annotated small ROIs and predict on large volumes.

SuRVoS2 core functionality is available entirely within the GUI for users who wish to interact without programming or scripting. Using the combination of tools available in SuRVoS2 to both annotate data and to train segmentation models, a domain expert can accelerate the process of segmenting their data and further can perform a variety of analyses from the output segmentations, all within the GUI. Additionally, for users familiar with scripting, the SuRVoS2 API and convenience functions allow for accessing and manipulating the entire process of segmentation within a SuRVoS2 workspace from within a Jupyter notebook. Importantly, the SuRVoS2 GUI can be run in parallel with Jupyter notebooks that are accessing the same workspace, providing an interactive analysis or development experience with access to 2D and 3D visualization options. Additionally, application developers can utilize the SuRVoS2 plugin system to extend SuRVoS2 functionality with their own custom plugins. This growing ecosystem within SuRVoS2 and Napari provides an exciting outlook for the future of volumetric image analysis and visualisation.

By separating SuRVoS2 into a client-server architecture, the hardware requirements and installation challenges of the software have been alleviated. The client, installed on a personal or work computer, requires minimal effort to install; and the server installation, which has also been simplified, can be completed by expert technicians on clusters and/or in the cloud to serve a large user base.

Future Development Plans

Now that the client-server architecture has been implemented, the immediate bottleneck associated with processing large datasets within SuRVoS2 has been alleviated, especially when the ROI-based deep learning pipeline is used. However, more efficient use of Dask cluster can be implemented alongside the use of Next Generation File Formats (Moore et al., 2020) to allow further parallelization of image processing operations which will allow SuRVoS2 to scale to even larger datasets.

A second area of development is around use of SuRVoS2 in the cloud. This has been tested with a limited set of virtual machines (Google cloud with Chrome Remote Desktop for client access). This configuration has provided good performance without loss of interactivity. In future, this will be supported through use of a SuRVoS2 server docker container, which could be hosted in the cloud and connected to an external client. This will also enable the possibility of using mobile and browser-based clients for further improvements to accessibility.

Third, SuRVoS2 will benefit from updates and improvements provided by the Napari plugin ecosystem. Napari provides a common user interface paradigm for all Napari plugins to use, easing the burden on researchers to learn bespoke plugins. And the functionality provided by new Napari plugins, especially in the area of segmentation tools, for example Stardist (Schmidt, et al., 2018) and ZELDA (D'Antuono and Pisignano, 2021) can be incorporated within a SuRVoS2 workflow.

A final area of research and development within the SuRVoS2 team is around the need for quantitative quality metrics embedded within the annotation and segmentation process. Segmentation of volumetric biomedical data using machine learning strategies often requires multiple iterative passes of annotation, segmentation, and quantitative evaluation to achieve high-quality results. It is often difficult or impossible to determine how much annotation is required and how detailed the annotation must be to achieve a particular segmentation result. Evaluation of the segmentation output is often manual and subjective and often completed in 2D, highlighting the need for quantitative metrics embedded within the segmentation pipeline to enable the evaluation of the segmentation output. In the future, we plan to explore active learning and collaborative learning techniques (Konyushkova et al., 2015; Luengo et al., 2016; Yang et al., 2017), which will aid the user in selecting regions for annotation to reduce model uncertainty. This deep learning aided segmentation strategy has the potential to further reduce the amount of manually annotated data through smarter selection of ROIs.

CONCLUSION

SuRVoS2 offers a unique set of tools that combine image processing and segmentation with the management of geometric information and the tools available in the Napari viewer. Its redesigned client-server architecture and newly implemented deep learning segmentation pipeline addresses the need to process ever larger datasets. The advanced features provided by the SuRVoS2 API allow users comfortable with scripting to interact through a Jupyter notebook while still accessing the interactive viewing tools. This ecosystem is designed for extensibility through plugin systems both within Napari and SuRVoS2. Together, these features allow for easy exploration of volumetric biomedical and correlative multi-modal data in both 2D and 3D followed by accelerated segmentation pipelines and analysis tools, accelerating data processing and analysis across modalities and scales.

DATA AVAILABILITY STATEMENT

Publicly available datasets were analyzed in this study. This data can be found here: EMPIAR 10562: https://www.ebi.ac.uk/empiar/EMPIAR-10562/ EMPIAR 10416: https://www.ebi.ac.uk/empiar/EMPIAR-10416/ S-BIAD19 (bioimage archive): https://www.ebi.ac.uk/biostudies/BioImages/studies/S-BIAD19?query=S-BIAD19 SuRVoS2 GitHub: https://github.com/DiamondLightSource/SuRVoS2.

AUTHOR CONTRIBUTIONS

MB, MD, and IL conceived of the SuRVoS Workbench software. IL and AP introduced the client-server architecture. AP integrated SuRVoS Workbench into the Napari plugin ecosystem and developed the PyTorch based image processing. OK and AP developed the deep learning pipeline and addressed feature and bug requests from beta-users. WT and MD made feature requests. WT was the main internal tester and trainer. EH Assisted in testing SuRVoS and created the training videos. AP, MD, and OK wrote the paper, all authors reviewed prior to submission.

FUNDING

This work was funded by Wellcome Trust grants 212980/Z/18/Z (AP, WT, MD, and MB) and 220526/Z/20/Z (EH, MD, and MB).

ACKNOWLEDGMENTS

We gratefully acknowledge the Stuart lab at the University of Oxford for providing unpublished test data. We acknowledge EMBL-EBI and the data repositories they facilitate, and those researchers who deposit data, which enables us to re-use previously published data to test new software and algorithms. We are also grateful to the following early adopters and testers of SuRVoS2: Matthew C. Spink, Andrew J. Bodey, Luis Perdigao, and Jennifer Pearson-Farr. This publication uses data generated *via* the Zooniverse.org platform, development of which is funded by generous support, including a Global Impact Award from Google, and by a grant from the Alfred P. Sloan Foundation.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcell.2022.842342/full#supplementary-material

REFERENCES

Abràmoff, M. D., Magalhães, P. J., and Ram, S. J. (2004). Image Processing with ImageJ. *Biophotonics Int.* 11 (7), 36–42.

Alvarez-Borges, F. J., King, O. N. F., Madhusudhan, B. N., Connolley, T., Basham, M., and Ahmed, S. I. (2021). U-Net Segmentation Methods for Variable-Contrast XCT Images of Methane-Bearing Sand. *Earth and Space Science Open Archive*. doi:10.1002/essoar.10506807.1

Basham (2021). GitHub - DiamondLightSource/SuRVoS2: Next Generation of SuRVoS. [online] Available at: https://github.com/DiamondLightSource/SuRVoS2 (Accessed December 13, 2021).

Belevich, I., and Jokitalo, E. (2021). DeepMIB: User-Friendly and Open-Source Software for Training of Deep Learning Network for Biological Image Segmentation. *Plos Comput. Biol.* 17 (3), e1008374. doi:10.1371/journal.pcbi.1008374

Bepler, T., Kelley, K., Noble, A. J., and Berger, B. (2020). Topaz-Denoise: General Deep Denoising Models for cryoEM and cryoET. *Nat. Commun.* 11 (1), 5208. doi:10.1038/s41467-020-18952-1

Berg, S., Kutra, D., Kroeger, T., Straehle, C. N., Kausler, B. X., Haubold, C., Schiegg, M., Ales, J., Beier, T., Rudy, M., Eren, K., Cervantes, J. I., Xu, B., Beuttenmueller, F., Wolny, A., Zhang, C., Koethe, U., Hamprecht, F. A., and Kreshuk, A. (2019). Ilastik: Interactive Machine Learning for (Bio)image Analysis. *Nat. Methods* 16 (12), 1226–1232. doi:10.1038/s41592-019-0582-9

Chan, T. F., and Vese, L. A. (2001). Active Contours without Edges. *IEEE Trans. Image Process.* 10 (2), 266–277. doi:10.1109/83.902291

Crosley, T. (2016). Hug: Embrace the APIs of the Future. [online] Hug. Available at: https://www.hug.rest (Accessed December 13, 2021).

D'Antuono, R., and Pisignano, G. (2022). ZELDA: A 3D Image Segmentation and Parent-Child Relation Plugin for Microscopy Image Analysis in Napari. *Front. Comput. Sci.*. doi:10.3389/fcomp.2021.796117

Darrow, M. C., Luengo, I., Basham, M., Spink, M. C., Irvine, S., French, A. P., et al. (2017). Volume Segmentation and Analysis of Biological Materials Using SuRVoS (Super-region Volume Segmentation) Workbench. *J. Vis. Exp.* 126, e56162. doi:10.3791/56162

Gibson, E., Li, W., Sudre, C., Fidon, L., Shakir, D. I., Wang, G., et al. (2018). NiftyNet: a Deep-Learning Platform for Medical Imaging. *Computer Methods Programs Biomed.* 158, 113–122. doi:10.1016/j.cmpb.2018.01.025

Gómez-de-Mariscal, E., García-López-de-Haro, C., Ouyang, W., Donati, L., Lundberg, E., Unser, M., et al. (2021). DeepImageJ: A User-Friendly Environment to Run Deep Learning Models in ImageJ. *Nat. Methods* 18, 1192–1195.

Haberl, M. G., Churas, C., Tindall, L., Boassa, D., Phan, S., Bushong, E. A., et al. (2018). CDeep3M-Plug-and-Play Cloud-Based Deep Learning for Image Segmentation. *Nat. Methods* 15 (9), 677–680. doi:10.1038/s41592-018-0106-z

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA Data Mining Software. *SIGKDD Explor. Newsl.* 11 (1), 10–18. doi:10.1145/1656274.1656278

Hand, D. J. (2020). *"Dark Data: Why what You Don't Know Matters" Princeton*. Princeton, NJ: University Press.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning. Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778.

Helmstaedter, M., Briggman, K. L., and Denk, W. (2011). High-accuracy Neurite Reconstruction for High-Throughput Neuroanatomy. *Nat. Neurosci.* 14 (8), 1081–1088. doi:10.1038/nn.2868

Howard, J., and Gugger, S. (2020). Fastai: a Layered API for Deep Learning. *Information* 11 (2), 108. doi:10.3390/info11020108

Jaeger, P. F., Kohl, S. S. A., Bickelhaupt, S., Isensee, F., Kuder, T. A., Schlemmer, H. P., et al. (2018). "Retina U-Net: Embarrassingly Simple Exploitation of Segmentation Supervision for Medical Object Detection," In Proceedings Machine Learning for Health NeurIPS Workshop, (PMLR), 171–183.

Khoreva, A., Benenson, R., Jan, H., Hein, M., and Bernt, S. (2017). "Simple Does it: Weakly Supervised Instance and Semantic Segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 876–885. doi:10.1109/cvpr.2017.181

King, O. (2021). DiamondLightSource/gas-hydrate-segmentation-unets: Code for Segmentation of 3-dimensional X-ray Computed Tomography (XCT) Images of Methane Bearing Sand Using 2d and 3d U-Net Models. (n.d.). RetrievedAvailable at: https://github.com/DiamondLightSource/gas-hydrate-segmentation-unets December 13, 2021).

Kluyver, T., Ragan-Kelley, B., Perez, F., Granger, B., Bussonnier, M., Frederic, J., et al. (2016). "Jupyter Notebooks-A Publishing Format for Reproducible Computational Workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Editors F. Loizides and B. Scmidt (Amsterdam, Netherlands: IOS Press), 87–90.

Konyushkova, K., Sznitman, R., and Pascal, F. (2015). "Introducing Geometry in Active Learning for Image Segmentation," in Proceedings of the IEEE International Conference on Computer Vision, 2974–2982.

Kremer, J. R., Mastronarde, D. N., and McIntosh, J. R. (1996). Computer Visualization of Three-Dimensional Image Data Using IMOD. *J. Struct. Biol.* 116 (1), 71–76. doi:10.1109/iccv.2015.340

Li, Q., Arnab, A., and Torr, P. (2018). "Weakly-and Semi-supervised Panoptic Segmentation," in Proceedings of the European conference on computer vision (ECCV), 102–118. doi:10.1007/978-3-030-01267-0_7

Lin, D., Dai, J., Jia, J., He, K., and Sun, J. (2016). "Scribblesup: Scribble-Supervised Convolutional Networks for Semantic Segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 3159–3167. doi:10.1109/cvpr.2016.344

Lin, T., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2016). Feature Pyramid Networks for Object Detection. Available at: https://arxiv.org/abs/1612.03144v2.

Lucchi, A., Smith, K., Achanta, R., Knott, G., and Fua, P. (2011). Supervoxel-based Segmentation of Mitochondria in Em Image Stacks with Learned Shape Features. *IEEE Trans. Med. Imaging* 31 (2), 474–486. doi:10.1109/TMI.2011.2171705

Luengo, I., Basham, M., and French, A. P. (2016). "Selective Labeling: Identifying Representative Sub-volumes for Interactive Segmentation," in International Workshop on Patch-based Techniques in Medical Imaging (Cham: Springer), 17–24. doi:10.1007/978-3-319-47118-1_3

Luengo, I., Darrow, M. C., Spink, M. C., Sun, Y., Dai, W., He, C. Y., et al. (2017). SuRVoS: Super-region Volume Segmentation Workbench. *J. Struct. Biol.* 198 (1), 43–53. doi:10.1016/j.jsb.2017.02.007

McInnes, L., Healy, J., Saul, N., and Groβberger, L. (2018). UMAP: Uniform Manifold Approximation and Projection. *J. Open Source Software* 3, 861.

McQuin, C., Goodman, A., Chernyshev, V., Kamentsky, L., Cimini, B. A., Karhohs, K. W., et al. (2018). CellProfiler 3.0: Next-Generation Image Processing for Biology. *Plos Biol.* 16 (7), e2005970. doi:10.1371/journal.pbio.2005970

Merkel, D. (2014). Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.* 239, 2.

Moore, J. (2020). Next-generation File Format (NGFF) Specifications for Storing Bioimaging Data in the Cloud. Open Microscopy Environment Consortium. This edition of the specification is Available at: The latest edition is Available at: https://ngff.openmicroscopy.org/0.1/https://ngff.openmicroscopy.org/latest/.

Napari Contributors (2021). GitHub - Napari/napari: Napari: a Fast, Interactive, Multi-Dimensional Image Viewer for python. [online] GitHub. Available at: https://github.com/napari/napari (Accessed December 13, 2021).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., et al. (2019). Pytorch: An Imperative Style, High-Performance Deep Learning Library. *Adv. Neural Inf. Process. Syst.* 32, 8026–8037.

Peddie, C. J., and Collinson, L. M. (2014). Exploring the Third Dimension: Volume Electron Microscopy Comes of Age. *Micron* 61, 9–19. doi:10.1016/j.micron.2014.01.009

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12, 2825–2830.

Perdigao, L. (2021). GitHub - rosalindfranklininstitute/RedLionfish. [online] Available at: https://github.com/rosalindfranklininstitute/RedLionfish/ (Accessed December 13, 2021).

Pérez-García, F., Sparks, R., and Ourselin, S. (2021). TorchIO: a Python Library for Efficient Loading, Preprocessing, Augmentation and Patch-Based Sampling of Medical Images in Deep Learning. *Comput. Methods Prog. Biomed.* 208, 106236. doi:10.1016/j.cmpb.2021.106236

Prigent, S. (2021). Napari Hub | Plugins | Napari-Stracking by Sylvain Prigent. [online] Napari-hub.org. Available at: https://www.napari-hub.org/plugins/napari-stracking (Accessed December 13, 2021).

Riba, E., Mishkin, D., Ponsa, D., Rublee, E., and Bradski, G. (2020). Kornia: An Open Source Differentiable Computer Vision Library for PyTorch, Winter Conference on Applications of Computer Vision. *IEEE Computer Society*, 3663–3672.

Riverbankcomputing.com (2021). Riverbank Computing | Introduction. [online] Available at: https://riverbankcomputing.com/software/pyqt/ (Accessed December 13, 2021).

Rocklin, M. (2015). "Dask: Parallel Computation with Blocked Algorithms and Task Scheduling," in Proceedings of the 14th python in science conference, 136.130

Ronneberger, O., Fischer, P., and Brox, T. (2015). "U-net: Convolutional Networks for Biomedical Image Segmentation," in International Conference on Medical image computing and computer-assisted intervention (Cham: Springer), 234–241. doi:10.1007/978-3-319-24574-4_28

Schmidt, U., Weigert, M., Broaddus, C., and Myers, G. (2018). "Cell Detection with star-convex Polygons," in International Conference on Medical Image Computing and Computer-Assisted Intervention (Cham: Springer), 265–273. doi:10.1007/978-3-030-00934-2_30

Seo, H., Badiei Khuzani, M., Vasudevan, V., Huang, C., Ren, H., Xiao, R., et al. (2020). Machine Learning Techniques for Biomedical Image Segmentation: An Overview of Technical Aspects and Introduction to State-Of-Art Applications. *Med. Phys.* 47 (5), e148–e167. doi:10.1002/mp.13649

Sofroniew, N. (2021). Napari Hub | Plugins | Napari-Animation by Nicholas Sofroniew, Alister Burt, Guillaume Witz, Faris Abouakil, Talley Lambert. [online] Napari-hub.org. Available at: https://www.napari-hub.org/plugins/napari-animation (Accessed December 13, 2021).

Strotton, M. C., Bodey, A. J., Wanelik, K., Darrow, M. C., Medina, E., Hobbs, C., et al. (2018). Optimising Complementary Soft Tissue Synchrotron X-ray Microtomography for Reversibly-Stained central Nervous System Samples. *Sci. Rep.* 8 (1), 12017–12018. doi:10.1038/s41598-018-30520-8

Sutton, G., Sun, D., Fu, X., Kotecha, A., Hecksel, C. W., Clare, D. K., et al. (2020). Assembly Intermediates of Orthoreovirus Captured in the Cell. *Nat. Commun.* 11 (11), 44451–44457. doi:10.1038/s41467-020-18243-9

Thermofisher.com (2021). Avizo Software for Materials Research | Thermo Fisher Scientific - UK. [online] Available at: https://www.thermofisher.com/uk/en/home/electron-microscopy/products/software-em-3d-vis/avizo-software.html (Accessed December 13, 2021).

Tun, W. M., Poologasundarampillai, G., Bischof, H., Nye, G., King, O. N. F., Basham, M., et al. (2021). A Massively Multi-Scale Approach to Characterizing Tissue Architecture by Synchrotron Micro-CT Applied to the Human Placenta. *J. R. Soc. Interf.* 18 (179), 20210140. doi:10.1098/RSIF.2021.0140

Van der Maaten, L., and Hinton, G. (2008). Visualizing Data Using T-SNE. *J. machine Learn. Res.* 9 (11).

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., et al. (2014). Scikit-image: Image Processing in Python. *PeerJ* 2, e453. doi:10.7717/peerj.453

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nat. Methods* 17 (3), 261–272. doi:10.1038/s41592-019-0686-2

Wagner, T., Merino, F., Stabrin, M., Moriya, T., Antoni, C., Apelbaum, A., et al. (2019). SPHIRE-crYOLO Is a Fast and Accurate Fully Automated Particle Picker for Cryo-EM. *Commun. Biol.* 2 (1), 218. doi:10.1038/s42003-019-0437-z

Wei, D., Lin, Z., Franco-Barranco, D., Wendt, N., Liu, X., Yin, W., et al. (2020). MitoEM Dataset: Large-Scale 3D Mitochondria Instance Segmentation from EM Images. *Int. Conf. Med. Image Comput. Computer-Assisted Intervention* 12265, 66–76. doi:10.1007/978-3-030-59722-1_7

Xu, C. S., Pang, S., Shtengel, G., Müller, A., Ritter, A. T., Hoffman, H. K., et al. (2021). An Open-Access Volume Electron Microscopy Atlas of Whole Cells and Tissues. *Nature* 599, 147–151. doi:10.1038/s41586-021-03992-4

Yang, L., Zhang, Y., Chen, J., Zhang, S., Chen, D. Z., and Chen (2017). "Suggestive Annotation: A Deep Active Learning Framework for Biomedical Image Segmentation," in *International conference on medical image computing and computer-assisted intervention*. Editors M. Descoteaux, L. Maier-Hein, A. Franz, P. Jannin, D. L. Collins, and S. Duchesne (Cham: Springer), 399–407. doi:10.1007/978-3-319-66179-7_46

Yoshiyuki, K., Jaerin, S., and Yasuo, K. (2018). Large Volume Electron Microscopy and Neural Microcircuit Analysis. *Front. Neural Circuits* 12. doi:10.3389/fncir.2018.00098

Zooniverse (2016). Available at: https://www.zooniverse.org/ (Accessed October 27, 2019).

Conflict of Interest: Authors AP, OK, WT, IL, and MB were employed by the company Diamond Light Source Ltd. EH was employed by the Rosalind Franklin Institute.

The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Copyright © 2022 Pennington, King, Tun, Ho, Luengo, Darrow and Basham. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.