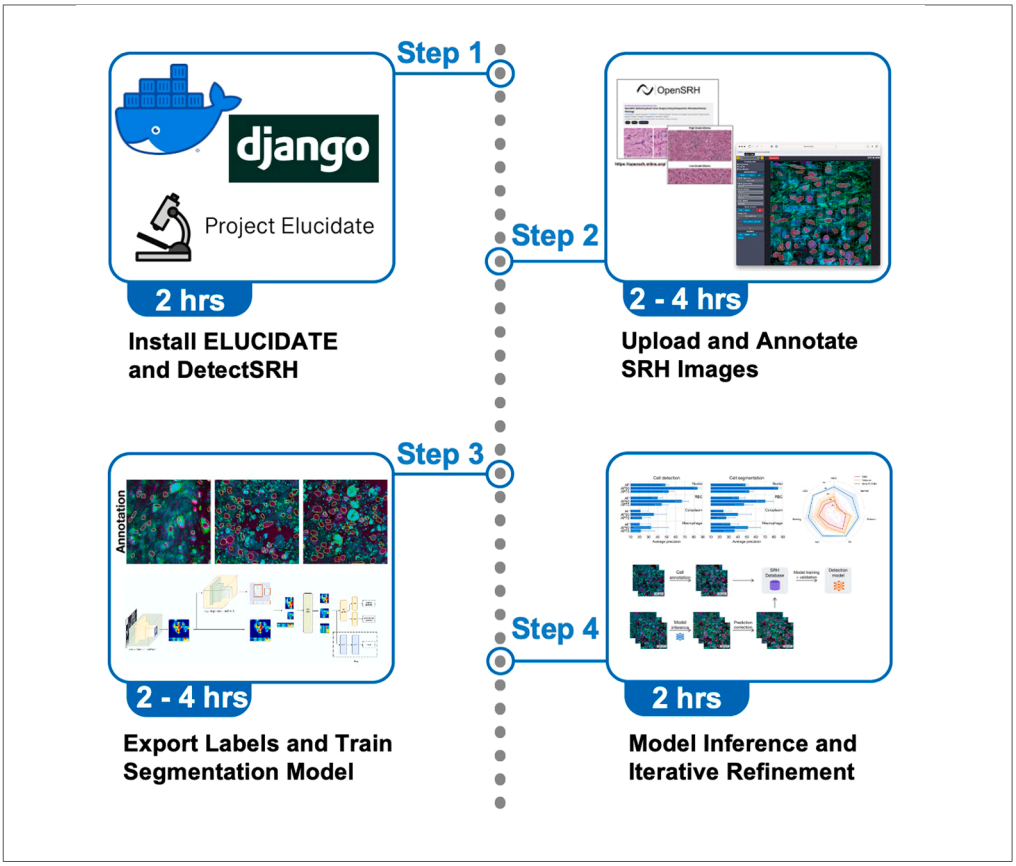
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol to annotate and automate single-cell instance segmentation on stimulated Raman histology using deep learning



Stimulated Raman histology (SRH) is a label-free optical imaging technique that can discern molecular components such as lipids and proteins at subcellular spatial resolution without histologic staining. Here, we present a protocol for labeling cells and training AI models for automated cell segmentation on SRH images acquired intra-operatively from neurosurgical cases. We describe steps to enable single-cell spatial analysis on SRH using ELUCIDATE, a web-based SRH cell annotation tool, and DetectSRH Python library.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Abhishek Bhattacharya, Eric Landgraf, Cheng Jiang, ..., Xinhai Hou, Lisa Walsh, Todd C. Hollon

abhishek.bhattacharya@nyulangone.org (A.B.)
elandg@umich.edu (E.L.)
chengjia@umich.edu (C.J.)
tocho@umich.edu (T.C.H.)

Highlights

Step-by-step workflow from manual labeling to automated cell detection and analysis

Instructions for annotating stimulated Raman histology (SRH) images with ELUCIDATE

Guidance on training deep learning cell segmentation models with DetectSRH library

Steps for model evaluation and iterative refinement via prediction correction and retraining

Bhattacharya et al., STAR Protocols 6, 104221 December 19, 2025
https://doi.org/10.1016/j.xpro.2025.104221

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol to annotate and automate single-cell instance segmentation on stimulated Raman histology using deep learning

Abhishek Bhattacharya,[1,2,3,4,5,*] Eric Landgraf,[1,3,4,*] Cheng Jiang,[1,4,*] Asadur Chowdury,[1] Akhil Kondepudi,[1] Lin Wang,[1] Edward S. Harake,[1] Xinhai Hou,[1] Lisa Walsh,[1] and Todd C. Hollon[1,*]

[1]University of Michigan, Ann Arbor, MI 48109, USA

[2]NYU Langone, New York, NY 10016, USA

[3]These authors contributed equally

[4]Technical contact

[5]Lead contact

*Correspondence: abhishek.bhattacharya@nyulangone.org (A.B.), elandg@umich.edu (E.L.), chengjia@umich.edu (C.J.), tocho@umich.edu (T.C.H.)
https://doi.org/10.1016/j.xpro.2025.104221

SUMMARY

Stimulated Raman histology (SRH) is a label-free optical imaging technique that can discern molecular components such as lipids and proteins at subcellular spatial resolution without histologic staining. Here, we present a protocol for labeling cells and training AI models for automated cell segmentation on SRH images acquired intra-operatively from neurosurgical cases. We describe steps to enable single-cell spatial analysis on SRH using ELUCIDATE, a web-based SRH cell annotation tool, and DetectSRH Python library.

BEFORE YOU BEGIN

Single cell analysis of cancer histology is important for understanding the tumor microenvironment, which can provide valuable insight into tumor heterogeneity, disease progression and therapeutic response. In central nervous system (CNS) brain tumor research, an advanced tissue imaging technique, Stimulated Raman Histology (SRH), has enabled rapid, label-free visualization of brain tumor biopsies, capturing sub-cellular details such as cytoplasmic and nuclear features by measuring the spectral signature of carbon-hydrogen bonds in lipids and proteins.[1] This method has the advantage of not requiring stains or molecular labeling prior to histologic image acquisition. Instead, a 2D spectral profile of the tissue is obtained by laser-based Stimulated Raman Spectroscopy (SRS) microscopes and processed computationally to resemble a hematoxylin & eosin (H&E) stained tissue. Clinical SRS microscopes combined with computer-vision-based Artificial Intelligence (AI) models are being used by neurosurgeons intraoperatively for near real-time tumor classification with high accuracy.[2] Recent papers have shown deep learning models trained on SRH can identify key genetic alterations useful for grading CNS tumors.[3] To encourage further development of machine learning models on the SRH platform, researchers have open-sourced a large dataset of SRH images of brain tumors with pathologic annotations called OpenSRH.[4] However, there still is no well-curated set of single-cell level annotations for SRH.

Innovation

While foundation deep learning models exist for automated cell segmentation on histology, they remain limited to conventional modalities such as light microscopy and generalize poorly to unique platforms such as SRH. To address this need, we provide four key contributions: 1. ELUCIDATE: a collaborative, web-based cell annotation tool for SRH 2. DetectSRH: a Python library for developing

and training deep learning cell segmentation models on SRH 3. SRH550: A benchmark dataset containing expert-annotated cell segmentations across multiple brain tumor types 4. Pre-trained Mask R-CNN weights optimized for SRH cell segmentation using the SRH550 dataset. In this protocol, we detail the complete pipeline from cell annotation to model training, and demonstrate our domain-specific model's performance compared to off-the-shelf cell segmentation models through a benchmark study.

Institutional permissions

The study has been approved by the Institutional Review Board (HUM00083059), with informed consent from each patient prior to SRH imaging.

Downloading and setting up ELUCIDATE

The fastest and easiest way to run ELUCIDATE is with Docker (link for download in key resources table). While it can be run locally without Docker, there may be compatibility issues with the required Python libraries and the operating system.

1. To begin, download the ELUCIDATE project using git and navigate to the project folder.

```
% git clone git@github.com:MLNeurosurg/Elucidate.git

% cd Elucidate && git checkout docker
```

⚠ CRITICAL: Ensure it is on the "docker" branch for the latest build.

2. Change the "DJANGO_EMAIL_BACKEND" variable in the *.django* file located in the *.envs/local/* folder to your desired email server backend and email address.

Note: This can be left as default as well. Below is a snippet from the *.django* file:

```
DJANGO_EMAIL_BACKEND=django_ses.SESBackend # default

DJANGO_DEFAULT_FROM_EMAIL="your_name <your@email.com>"
```

3. Build the Docker containers/services and Postgres Database Container using the following commands from the root of the folder:

```
% docker-compose -f local.yml build --no-cache

% docker-compose -f local.yml run --rm django python manage.py migrate
```

4. Create a superuser account with the following command:

```
% docker-compose -f local.yml run --rm django python manage.py create superuser
```

Launching ELUCIDATE

5. Start the ELUCIDATE web service with the following command and then navigate to 'localhost:8000'.

```
% docker-compose -f local.yml up
```

Downloading and installing DetectSRH Python library

The DetectSRH Python package is a helpful library for training segmentation models on SRH using annotation data from ELUCIDATE. We recommend having MiniConda, a Python package and environment manager, installed prior to downloading DetectSRH.

6. Locally download the DetectSRH repository from GitHub.

```
% git clone git@github.com:MLNeurosurg/detectsrh.git detect_srh

% cd repo_name

% pip install -e .
```

7. Create and activate the Conda environment.

```
% conda create -n ds python=3.10

% conda activate ds
```

8. Install DetectSRH in the Conda environment.

```
% cd detect_srh

% pip install -e .
```

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
Python (v.3.8)	Python Software Foundation	https://www.python.org
MiniConda	Anaconda Inc.	https://docs.anaconda.com/miniconda/
Django (v.1.5)	Django Project	https://www.djangoproject.com/
Cookiecutter Django	Greenfield et al.	https://github.com/cookiecutter/cookiecutter-django
Docker	Docker Inc.	https://www.docker.com/
Django-Labeller	French et al.	https://github.com/Britefury/django-labeller
Pytorch	Paszke et al.	https://pytorch.org/
Scikit Learn	Buitinck et al.	https://doi.org/10.48550/arXiv.1309.0238
Pandas	The pandas development team	https://doi.org/10.5281/zenodo.3509134
Numpy	Harris et al	https://doi.org/10.1038/s41586-020-2649-2.
Matplotlib	Hunter et al.	https://doi.org/10.1109/MCSE.2007.55
Deposited data		
ELUCIDATE Web Platform GitHub repository	This paper	https://github.com/MLNeurosurg/elucidate
DetectSRH GitHub repository	This paper	https://github.com/MLNeurosurg/detectsrh
SRH550 Dataset	This paper	https://mlins.org/elucidate/
Mask R-CNN Model Weights	This paper	https://mlins.org/elucidate/

MATERIALS AND EQUIPMENT

This protocol was developed and tested across multiple compute platforms for both ELUCIDATE deployment and DetectSRH model training. ELUCIDATE web platform can run at minimum on a laptop for single users, on a desktop with local area network (LAN) access for small teams (2-3 users)

in laboratory environments, or on cloud infrastructure such as Amazon Web Services (AWS) for multi-site collaborations (3+ users). DetectSRH requires at minimum a CUDA-capable NVIDIA GPU with 8GB VRAM. We conducted DetectSRH experiments using both single GPU workstations and a HIPAA-compliant Linux high-performance computing (HPC) cluster at the University of Michigan with GeForce RTX 2080 Ti GPUs.

We outline minimum, recommended, and optimal computing specifications in the tables below as a guide for users to determine which deployment scale is appropriate for their needs.

ELUCIDATE web platform	Minimum (single user)	Recommended (2–3 users)	Optimal (3+ users)
CPU	Intel i5-8400 or equivalent	Intel i7-10700, AMD Ryzen 7 3700×, or Apple M1 Pro/Max	AWS EC2 t2.medium or greater, scalable
RAM	8 GB	16GB	4 GB+
Storage	64 GB free	1TB NVMe SSD	S3 + EBS storage
OS	Ubuntu 20.04, macOS 12+	Ubuntu 20.04, macOS 12+	Ubuntu 20.04 LTS
Network	Basic broadband	Ethernet connection preferred	High-Speed institutional network

DetectSRH	Minimum	Recommended	Optimal
GPU	Nvidia GTX 1080 (8 GB VRAM)	Nvidia RTX 2080Ti (11 GB VRAM)	HPC cluster (A100 40 GB+)
CPU	Intel i7-9700K or AMD Ryzen 5 3600×	Intel i9-10900K or AMD Ryzen 9 3900×	Xeon Gold 6248+ or EPYC 7742+
RAM	32 GB	64 GB	128 GB+
Storage	500 GB NVMe SSD	1TB NVMe SSD	Distributed storage system

STEP-BY-STEP METHOD DETAILS

Here, we provide step-by-step instructions for using the ELUCIDATE web-platform to annotate single cells on SRH images and training a segmentation model on the labeled data using DetectSRH Python library. We also show steps for correcting the model predictions in ELUCIDATE for sequential re-training and improvement of the model segmentation accuracy.

Upload SRH images

Timing: 10 min to 2 h

This section describes steps for uploading SRH images to the ELUCIDATE web platform for processing and labeling.

1. Go to the **Datasets** view and click **Add** on the top right corner (Figure 1).
2. Enter a dataset name (if dataset name already exists, images will be added to this set).
3. Ensure image type and filename are in the correct format (required).

⚠ CRITICAL: The ELUCIDATE web platform requires SRH images to be in Tagged Image File Format (TIFF) with 300×300 pixel dimensions and 2-channels.

⚠ CRITICAL: The images must be uploaded in a single folder as a .zip file with the naming scheme and preferred folder structure shown below. If there are pre-existing corresponding JSON label files, these can be placed in an "annotations" subdirectory. The JSON labels must have the *same filename* as the corresponding image.

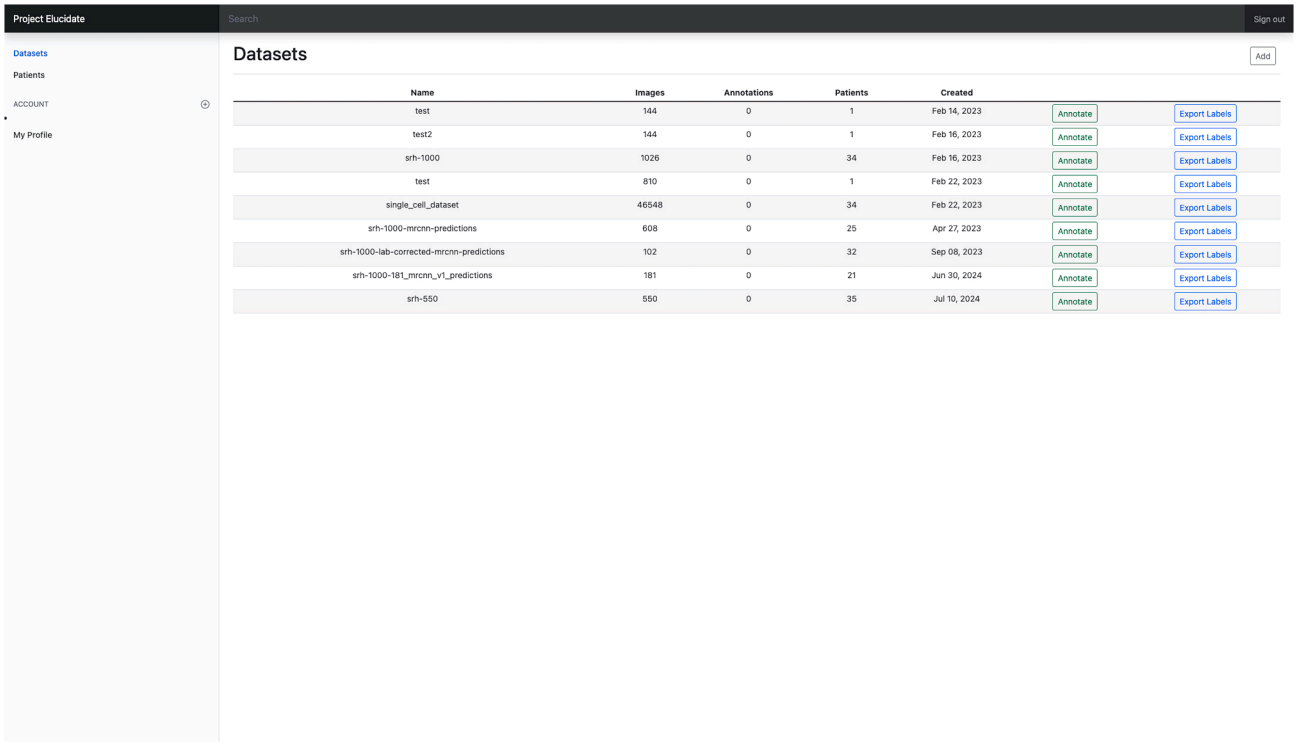


Figure 1. ELUCIDATE Dataset View

This is the initial view shown after logging into ELUCIDATE web platform. Shows a list of datasets created, corresponding metadata, and buttons to access the annotation viewer or to export annotated labels.

```
uploaded_folder.zip /
  [patient_id]-[slide_id]-file0.tiff
  [patient_id]-[slide_id]-file1.tiff
  [patient_id]-[slide_id]-file2.tiff
  ...
  annotations/
    [patient_id]-[slide_id]-file0.json
    [patient_id]-[slide_id]-file1.json
    [patient_id]-[slide_id]-file2.json
    ...
```

⚠ CRITICAL: The *patient_id* can be any alpha-numeric value but *slide_id* must be a numerical integer value.

4. Choose zip file from the file browser for upload and click **Upload** (Figure 2).

Note: All images are associated with an overarching Dataset (defined by you) and a Patient (automatically derived from image filename or metadata JSON file).

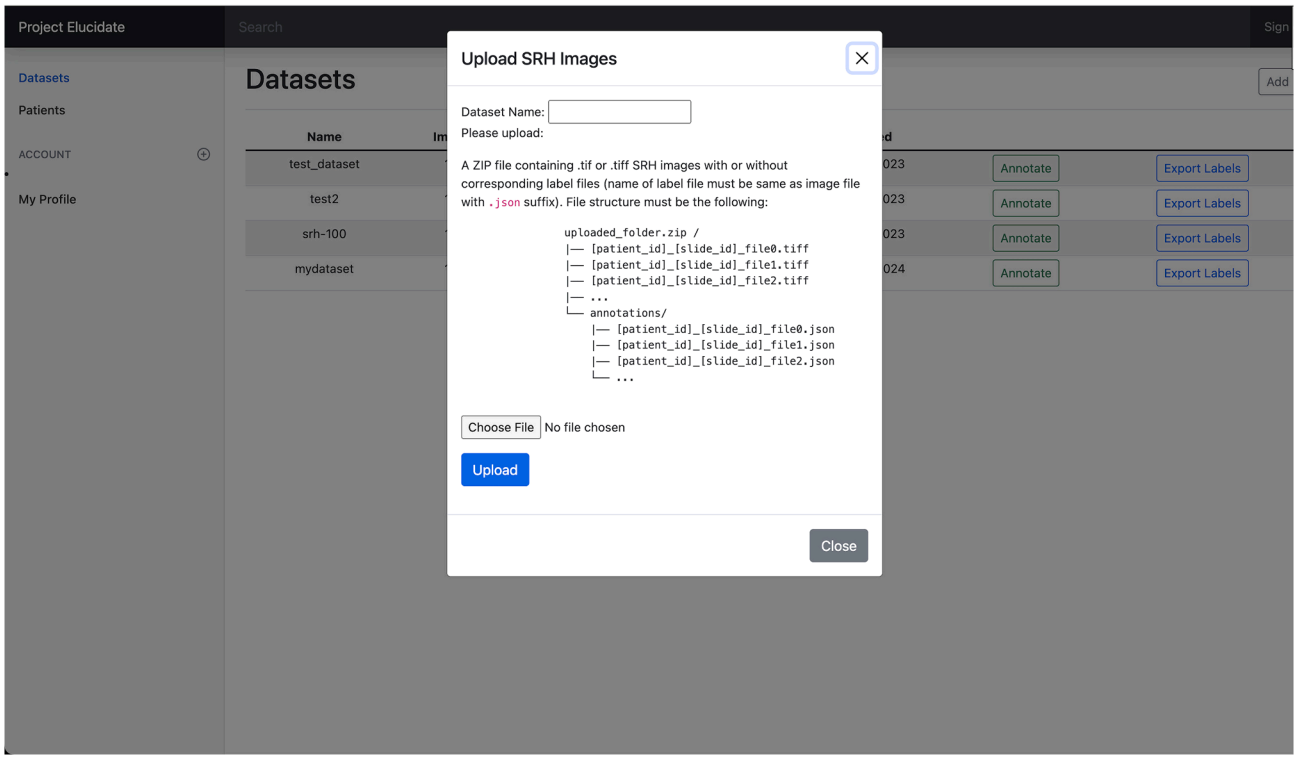


Figure 2. ELUCIDATE Upload View
Modal for creating a new dataset and uploading corresponding SRH images and labels to the platform.

Note: Upload time varies significantly with dataset size and network speed. Large datasets (10,000+ images) or slower internet connections may require 1 to 2 h or longer. Consider splitting large uploads into smaller batches for better reliability.

Create annotation labels

Timing: 10–15 min

Before starting the annotation process, labels need to be added through the ELUCIDATE web administration panel. The following steps shows how to create labels and choose the appropriate label class for each segmentation structure created in the annotation viewer.

5. Navigate to the administration panel by going to the following "[ELUCIDATE Web Url]/admin" url. Example view shown in Figure 3.

   Note: "VIEW SITE" link in the top right corner will return you to the non-admin website.

6. Create a **Label Class Group**.
   a. Choose the **Label Class Group** button below the "Image Labelling Tool" heading on the left-side menu (Figure 3).
   b. Click **Add Label Class Group** on the upper right hand corner.
   c. Choose 'default' for the Schema. If it does not exist, please create a Schema called 'default.'
   d. Define the name of the label class group.
   e. Provide an Order Index number to show in which order the label class groups should display in the annotation viewer.
   f. Repeat this step for each new label class group.

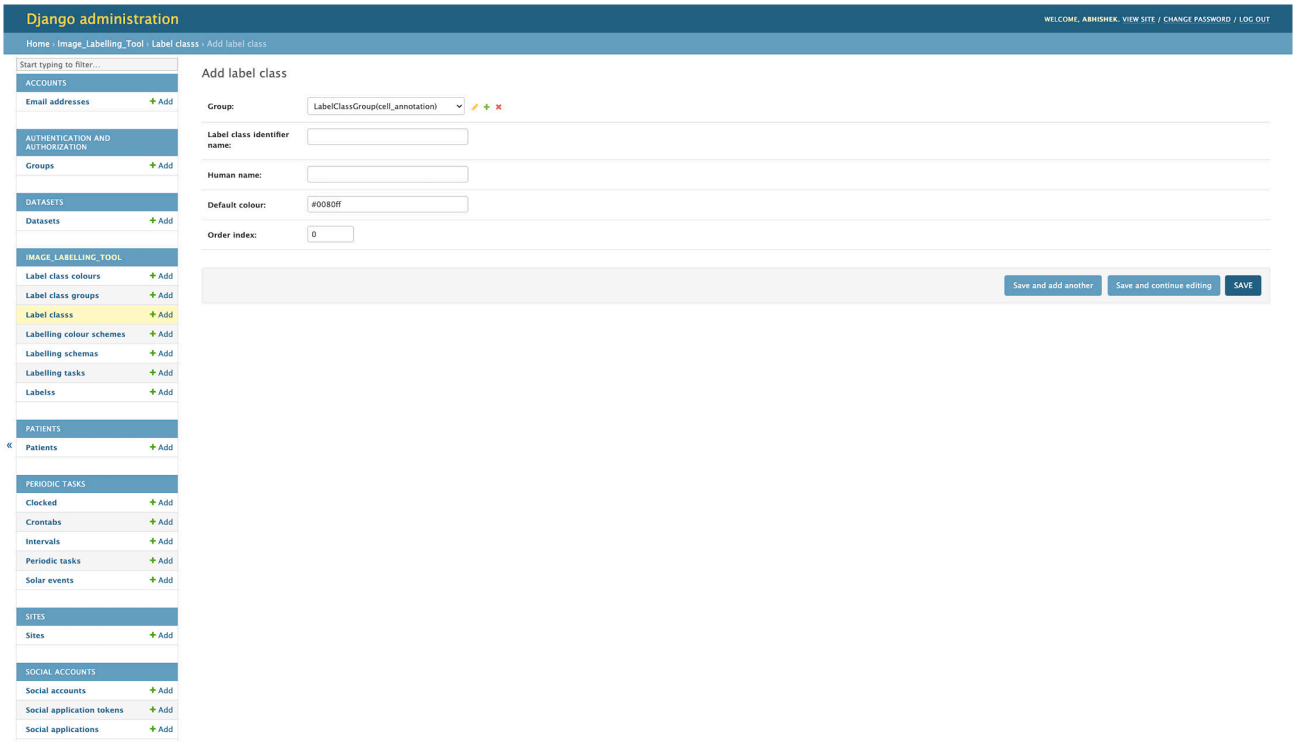


Figure 3. ELUCIDATE administration panel label group creation view

View of the administration panel for customizing various parameters on the web platform. This specific view shows the form for creating new label group.

⚠ CRITICAL: At least one Label Class Group is required to create labels.

Note: The **Label Class Group** allows for creating a set of labels under a specific category. For example, you can define a set of labels for "intra-cellular" structures such as "nuclei", "cytoplasm", etc. And also have another set named "non-cellular" which have labels such as "blood vessels", "axons", etc. Both will display on the tool panel when in the annotation viewer.

⚠ CRITICAL: Create Label Schema with name "default" if it has not already been defined.

7. Create a **Label Class**.

Note: These are the individual labels that can be applied to your cell segmentation annotations.

a. Choose the **Label Class** button below the "Image Labeling Tool" heading on the left-side menu.
b. Click **Add Label Class** on the upper right hand corner.
c. Choose a "Label Class Group" from the drop down menu as defined in the previous step.
d. Define a "Label Class Identifier" name (this is what will be defined in the exported labels in JSON).
e. Define a "Human Name" for the label class which will display on the annotation viewer.
f. (Optional) Choose a color for the label segmentation (values defined in hex).
g. Provide an Order Index number to show in which order the label class groups should display in the annotation viewer.
h. Repeat this step for each new label.

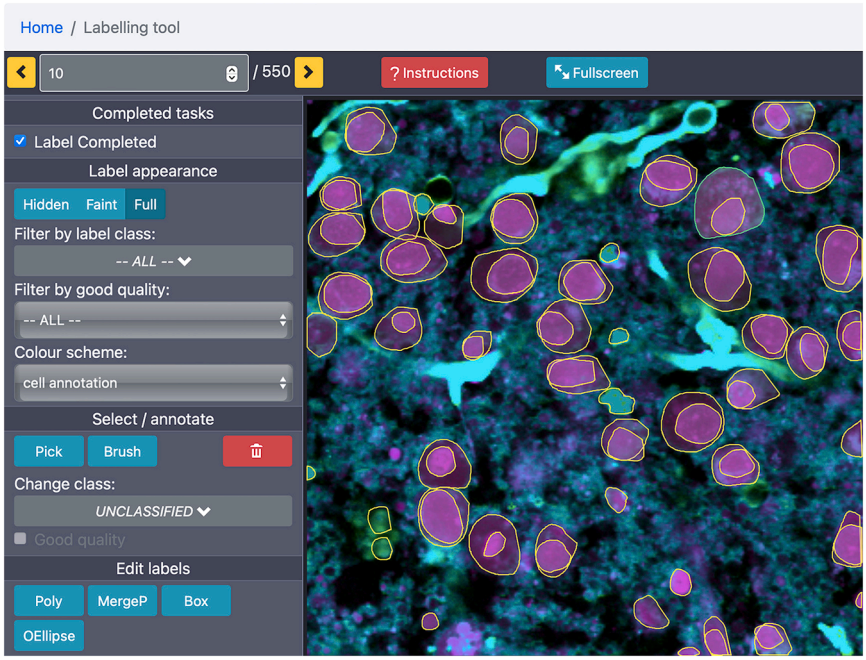


Figure 4. ELUCIDATE annotation interface with example cell segmentations

Annotate SRH images

Timing: 10 min to 2 h

SRH images can be viewed at the Dataset or Patient level. Each row in the Dataset or Patient table has an **Annotate** button that will pull up the corresponding set of SRH images for annotation. Once in the annotation viewer (Figure 4), you can create polygonal segmentations with assigned labels over any part of the image. The x, y coordinates with associated labels will be saved automatically and available for export. (Export instructions defined at a later step).

Note: Allow some time for images to load if the set has many images.

8. From the ELUCIDATE home page, choose either "Dataset" or "Patient" on the left-side navigation panel to see the list of SRH images by dataset or patient level.
9. Click the **Annotate** button on the right side of each dataset or patient row to open the annotation viewer.

Note: The annotation viewer will show the first SRH image in the set and you can use the yellow "<" and ">" buttons to tab through the images in the set.

10. Create a segmentation label for a cell or another structure in the SRH image.
    a. Ensure the "Brush" and "Poly" options are selected in the left-side tool panel.
    b. Select a label for the segmentation from the drop-down menu under "Change Class." Default option is "Unclassified."
    c. Hover over a cell structure (e.g., the cytoplasm) in the SRH image and then hold "Shift + left click" while drawing an outline. Can right-click when the outline is complete.
    d. Repeat 9b and 9c until all desired structures have been labeled. Same can be done for all subsequent images in the set.

Note: Segmentations can be deleted by choosing the "Pick" option on the left-side tool panel, then selecting the segmentation with left-click, followed by pressing the red "Trash" button.

Note: You can pan around the image by right-clicking and moving the blank space on the outside in the desired direction.

Note: Detailed instructions for how to use the other features in the annotation viewer can be seen by clicking the red **Instructions** button at the top of the interface.

Export annotations

Timing: 5 min

11. Once labeling is complete, navigate to the ELUCIDATE home page at the "Dataset" or "Patient" view.
12. Select the **Export Label** button next to the set that was annotated. This will initiate a download of a JSON file.

Note: The JSON file with annotation labels is formatted in the following manner:


```
[
  {
    "label_type": "polygon",
    "object_id": "7255478f-cf81-4102-ac79-4d0b8dcc6eba",
    "label_class": "cell_nuclei",
    "source": "manual",
    "anno_data": {},
    "regions": [
      [
        {
          "x": 230.0,
          "y": 129.0
        }, ...
      ]
    ]
  }, {
    "label_type": "polygon",
    "object_id": "0a3fe643-9a08-473d-bea9-8248a6f59f51__1",
    "label_class": "cytoplasm",
    "source": "manual",
    "anno_data": {},
    "regions": [
      [
```

```
      {
        "x": 116.0,
        "y": 229.0
      }, ...
    ]
  ]
 }, ...
]
```

Train, evaluate, and inference with a segmentation model

Timing: 30 min to 2 h

The DetectSRH Python library can be used to train a Mask R-CNN model on cell annotations exported from ELUCIDATE. The following instructions demonstrate model training, evaluation, and inference using the SRH550 data, which contains expert-annotated segmentations of brain tumor biopsies labelled with the ELUCIDATE web platform. The following code can be adapted for different datasets and segmentation models.

⚠ CRITICAL: Ensure DetectSRH Python library is installed in your MiniConda environment. Details for installation are described in the before you begin section. Link to the SRH550 dataset can be found in the key resources table.

13. Organize the SRH images and corresponding annotation labels in a directory structure matching the SRH550 dataset format as described in Table 1. Example directory structure shown below:

```
srh550/
 images/
  srh550_000-3-0_0_0_0.tif
  srh550_000-3-0_0_0_300.tif
  srh550_000-3-0_0_0_600.tif
  srh550_000-3-0_0_300_0.tif
    ... # other image files
 labels/
  srh550_000-3-0_0_0_0.json
  srh550_000-3-0_0_0_300.json
  srh550_000-3-0_0_0_600.json
  srh550_000-3-0_0_300_0.json
    ... # other annotation files
 srh550_meta.csv
```

14. Review the metadata CSV file. The SRH550 dataset includes a master CSV file (*srh550_meta.csv*) that contains the columns shown in Table 2.

Table 1. SRH550 Dataset contents

File/directory	Description
srh550	Root directory for dataset
images	Contains SRH images
labels	Contains exported JSON annotation files (must have same name as corresponding SRH image file)
srh550_meta.csv	CSV (comma-separated values) file containing metadata for all images in the directory. Details discussed in the next step.

Note: The variables within each column can be arbitrarily defined for your specific experiments. Below is an abbreviated view of the *srh550_meta* CSV file containing the image, patient, series, and fold columns.

```
image,patient,series,class,fold

srh550_000-3-0_1000_300_300.tif,srh550_000,3,hgg,2

srh550_000-3-0_0_300_600.tif,srh550_000,3,hgg,2

srh550_000-3-0_3000_300_300.tif,srh550_000,3,hgg,2

... # more patches from srh500_000

srh550_001-1-1000_2000_300_300.tif,srh550_001,1,normal,4

srh550_001-1-1000_4000_0_300.tif,srh550_001,1,normal,4

srh550_001-1-1000_2000_600_600.tif,srh550_001,1,normal,4

... # more patches from srh500_001

srh550_002-3-0_1000_300_0.tif,srh550_002,3,lgg,0

srh550_002-3-0_0_0_300.tif,srh550_002,3,lgg,0

srh550_002-3-0_2000_300_600.tif,srh550_002,3,lgg,0

... # more patches from srh500_001, and other patients
```

Note: The fold assignments in the CSV ensure a balanced distribution of tumor types while maintaining patient-level separation across the folds.

15. Configure parameters for model training.
    a. Navigate to the training configuration sub-directory in *detectsrh*.

```
% cd detectsrh/ds/train/config
```

    b. Update the infrastructure configurations in the *train_mrcnn.yaml* file, as shown in Table 3 Training infrastructure subsection.
    c. Update the data configurations in the *train_mrcnn.yaml* file, as shown below in Table 3 Dataset/dataloader subsection.
    d. Define training schedule, optimizer and model configurations in the *train_mrcnn.yaml* file, as shown below in Table 3 Model training subsection.

```
# train_mrcnn.yaml

# only key parameters illustrated, full configuration file available in the released repository

infra:

  exp_name: srh_segmentation

  comment: 5fold_training
```

Table 2. *srh550_meta.csv* column descriptions

Metadata CSV *column*	Description
image	Filename of the SRH image
patient	Patient identifier (e.g., srh550_000, srh550_001)
series	Series identifier (e.g., 3) – this column can be omitted as it refers to the sequence of the specific image in the larger whole slide image collection
class	Diagnostic class of the biopsied tumor (e.g., high grade glioma – hgg, low grade glioma – lgg, normal, etc.)
fold	Assigned fold number (0-4 for 5-fold cross-validation). Used for splitting data for balanced training and evaluation.

```
  log_dir: /path/to/experiments
data:
  which: SRHSingleCell
  direct_params:
    common:
      data_root: /path/to/srh550
      slides_file: /path/to/srh550_meta.csv
    train:
      folds: [0,1,2,4]
    val:
      folds: [3]
model:
  name: mrcnn
training:
  num_epochs: 20
  optimizer:
  which: adamw
  params:
    lr: 2.0e-4
    weight_decay: 0.0001
```

Note: The configuration excerpt above illustrates key parameters in the *train_mrcnn.yaml* file. All parameters are described in full on GitHub. The path to the file is *ds/train/config/train_mrcnn.yaml*.

16. Run training experiment.
    a. Execute training with the following commands:

```
% cd detectsrh/ds/train
% python train.py -c=config/train_mrcnn.yaml
```

    b. Monitor training progress using Tensorboard as shown below.

Table 3. Training configuration file parameters

Parameter	Description
Training infrastructure	
infra.exp_name	The name of the experiment
infra.comment	The experiment instance (or experiment run)'s name will be the comment prepended with a unique identifier that includes the time the experiment was run and the *infra.seed*
infra.log_dir	Absolute path to where log outputs will be saved
infra.seed	Seed used for random number generation during training
Dataset/dataloader	
data.direct_params.common.data_root	Absolute path to root directory containing images and labels from Step 13
data.direct_params.common.slides_file	Absolute path to CSV containing assigned fold column
data.direct_params.train.folds	Integer array containing the folds to be used for training (e.g., [0, 1, 2, 4])
data.direct_params.val.folds	Integer array containing the folds to be used for validation (e.g., [3])
Model training	
training.num_epochs	Total training epochs (default: 20)
training.optimizer.which	Type of optimizer (default: adamw; other options include sgd)
training.optimizer.params.lr	Initial learning rate (default: 2.0e-4)
valid.freq.interval	Validation frequency (default: 10)
valid.freq.unit	Validation frequency unit (default: epoch)

```
% cd <infra.log_dir>/<infra.exp_name>/<exp_instance_name>

% tensorboard --logdir .
```

Note: Each experiment instance (or each experiment run)'s name will be prepended with a unique identifier that includes the time the experiment was run.

17. Configure parameters to evaluate model performance.
    a. Navigate to the evaluation configuration sub-directory in *detectsrh*.

```
% cd detectsrh/ds/eval/config
```

    b. Update the infrastructure configurations in the *eval_mrcnn.yaml* file. Parameters should be the same as the ones from the *train_mrcnn.yaml* file.
    c. Update the data configurations in the *eval_mrcnn.yaml* file. Parameters should be the same as the ones from the *train_mrcnn.yaml* file shown in Table 3.
    d. Specify the model checkpoint configuration in the *eval_mrcnn.yaml* file, as shown below in Table 4 Evaluation model checkpoint subsection.

```
# eval_mrcnn.yaml

# only key parameters illustrated, full configuration file available in the released repository

infra:

  exp_name: srh_segmentation

  comment: 5fold_training

  log_dir: /path/to/experiments

data:

  which: SRHSingleCell

  direct_params:
```

Table 4. Evaluation configuration file parameters

Parameter	Description
Evaluation infrastructure	Use the same values as in *train_mrcnn.yaml*
Dataset/dataloader	Use the same values as in *train_mrcnn.yaml*
Evaluation model checkpoint	
eval.ckpt_path	Relative path to model checkpoint from training – it should be in this form: *<log_dir>/<exp_name>/<instance_name>/models/<model_ckpt>.pth*

```
    common:
      data_root: /path/to/srh550
      slides_file: /path/to/srh550_meta.csv
    train:
      folds: [0,1,2,4]
    val:
      folds: [3]
model:
  name: mrcnn
eval:
  ckpt_path: relative/path/to/checkpoint.ckpt
```

Note: The configuration excerpt above illustrates key parameters in the *eval_mrcnn.yaml* file. All parameters are described in full on GitHub. The path to the file is *ds/eval/config/eval_mrcnn.yaml*.

18. Run model evaluation.
    a. Execute the evaluation script.

```
% cd detectsrh/ds/eval
% python eval.py -c=config/eval_mrcnn.yaml
```

    b. Results from the evaluation run are saved to the path below. Can navigate to this directory after evaluation is complete.

```
% cd <log_dir>/<exp_name>/<exp_instance_name>/evals/<run>/
```

    c. Review evaluation metrics saved in the *results/metrics.csv* file and predicted image segmentations in the *predictions* subdirectory.
19. Configure parameters for model inference.
    a. Navigate to the evaluation configuration sub-directory in *detectsrh*.

```
% cd detectsrh/ds/eval/config
```

    b. Update the inference configurations in the *inference_mrcnn.yaml* file shown in Table 5.

```
# inference_mrcnn.yaml
# only key parameters illustrated, full configuration file available in the released repository
ckpt_path: /path/to/checkpoint.ckpt
```

Table 5. Inference configuration file parameters

Parameter	Description
ckpt_path	Absolute path to model checkpoint
classes	List of classes model is trained to predict
confidence	Detection confidence threshold – this is for visualization purposes, and the threshold is applied after matrix non-maximum suppression reweighting (default: 0.50)
inference_dir	Absolute path to directory containing images to run inference on
out_dir	Absolute path to output directory for model prediction visualizations and annotations

```
classes: [na, nuclei, cyto, rbc, mp]

confidence: 0.5

inference_dir: /path/to/patches

out_dir: /path/to/output
```

Note: The configuration excerpt above illustrates key parameters in the *inference_mrcnn.yaml* file. All parameters are described in full on GitHub. The path to the file is *ds/eval/config/inference_mrcnn.yaml*.

20. Run model inference
    a. Execute the inference script.

```
% cd detectsrh/ds/eval

% python inference.py -c=config/inference_mrcnn.yaml
```

    b. Results from the inference run are saved to the path below. Can navigate to this directory after inference is complete.

```
% cd <out_dir>
```

    c. Review *result.pt* and *annotations* directory.

Note: There is also an interactive Jupyter notebook provided for running inference on images, located at *playgrounds/inference.ipynb*. It is ideal for visualizing select SRH images with single cell annotations and model predictions.

Prediction correction and iterative model refinement

Timing: 1–2 h

The integration of ELUCIDATE and the DetectSRH library through standardized data formats enables iterative model refinement as one accumulates both manually annotated and model-generated annotations. This approach leverages model predictions as a starting point for expert correction, significantly reducing annotation time while maintaining quality. The corrected annotations then serve as an expanded training dataset for subsequent model iterations. The following streamlined workflow enables the potential for continuous improvement of segmentation accuracy through model-assisted annotation. Graphical representation of the workflow is shown in Figure 5.

21. Train baseline Mask R-CNN model using DetectSRH library on manually annotated SRH images as shown in Steps 13-16.

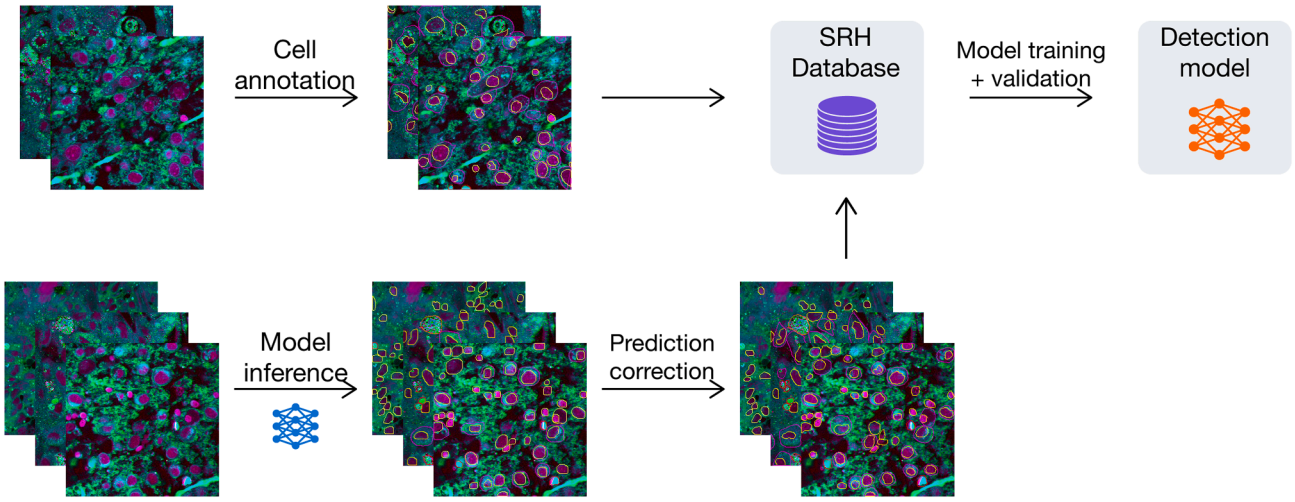


Figure 5. Building SRH Annotation Database Using Model-Assistance

The diagram illustrates the manual (above) and model-assisted methods (below) workflow for building a large SRH database of cell annotations. The workflow below allows for correcting model predictions directly on ELUCIDATE before feeding it in for model training and validation, which can be repeated as needed.

22. Using this model, run inference on unlabeled SRH images to generate JSON files containing predicted segmentation labels. Shown in Steps 19-20.
23. Compress prediction JSON files with corresponding images into a zip file for upload onto the ELUCIDATE web platform following the instructions from the upload SRH Images section.
24. Review predictions in the ELUCIDATE annotation interface and correct false positives/negatives. Add missed cell segmentations following the annotate SRH Images section.
25. Export the corrected annotations and combine them with original training data. Repeat training of the model with the expanded dataset.

Note: In summary, the iterative cycle follows these core steps: 1. Run inference using trained model 2. Upload predictions as a zip file to ELUCIDATE 3. Correct predicted annotations on ELUCIDATE platform 4. Export corrected predictions and combine with original data 5. Re-train model on expanded dataset.

EXPECTED OUTCOMES

Successful implementation of this protocol results in three major assets. First, users will have a fully functional ELUCIDATE web platform that enables collaborative cell annotation and dataset management. Second, following our annotation steps results, one should have a well-curated dataset of cell segmentations on SRH images. This organized collection of annotations provides the foundation for training deep learning segmentation models and serves as a template for researchers to develop their own SRH datasets. Third, researchers gain access to a deep learning-based cell segmentation model fine-tuned for SRH image analysis. This model generates high-quality instance segmentation predictions on SRH that can be further refined through the iterative correction process defined in this protocol.

Researchers following this protocol can readily apply these tools to analyze SRH images from the publicly available OpenSRH repository. The combination of collaborative annotation tools, structured datasets, and optimized models provides a complete framework for advancing computational pathology research and quantitative single-cell analysis of CNS malignancies using SRH imaging.

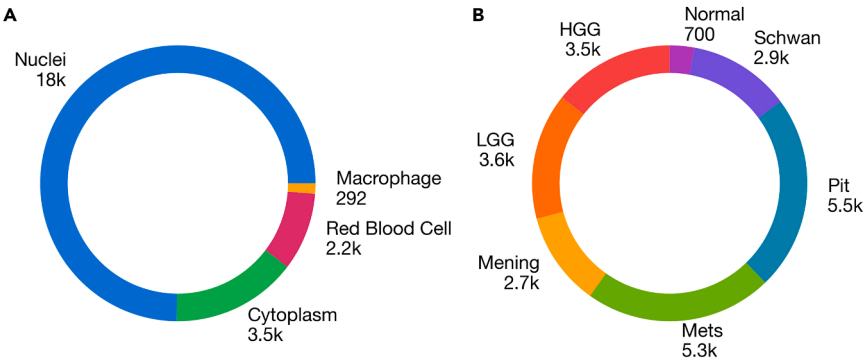


Figure 6. Distribution of cell annotations

(A) Number of annotations per cell type or cell structure label.

(B) Number of annotations per tumor type. Abbreviations: HGG, high grade glioma; LGG, low grade glioma; Mening, meningioma; Mets, metastasis; Pit, pituitary adenoma; Schwan, schwannoma; Normal, normal brain.

QUANTIFICATION AND STATISTICAL ANALYSIS

For the purposes of this protocol, we launched ELUCIDATE on a cloud computing provider, Amazon Web Services, using Docker. We uploaded a dataset of 550 SRH patch images from the OpenSRH repository, which includes samples from thirty-five de-identified University of Michigan patients, covering six brain tumor types – high grade glioma, low grade glioma, meningioma, metastasis, pituitary adenoma, and schwannoma—as well as normal brain tissue. A team of three optical imaging experts annotated a subset of 369 patch images, focusing on cell nuclei, cytoplasm, red blood cells, and macrophages. To showcase the iterative correction process, we trained a Mask R-CNN model on the 369 images and then inferenced on the remaining 181 images. The predicted annotations on the latter set were then corrected using the ELUCIDATE platform. The final 550 SRH set was reviewed and verified by a board-certified neurosurgeon. A breakdown of the number of annotations for each cell type or cell structure is shown in Figure 6A and the number of annotations per tumor type is shown in Figure 6B.

To validate our pipeline's effectiveness in developing robust cell segmentation models on SRH, we conducted a benchmarking experiment comparing predictions from a Mask R-CNN model provided through our DetectSRH library with two leading segmentation methods: Cellpose, a specialized deep learning model for biological image segmentation,[5] and the Segment Anything Model (SAM), a foundation model for general-purpose image segmentation.[6] We trained a Mask R-CNN following the protocol in the Train and Evaluate Segmentation Model section and set training parameters as described in Table 6. We performed 5-fold cross validation on the dataset with the distribution shown in Figure 7.

To evaluate the quality of the predicted labels against ground truth annotations, we use the average precision (AP) metric, which is widely adopted in object detection and instance segmentation evaluation tasks. Here, AP summarizes a precision-recall curve, where precision and recall values are computed after each predicted cell segmentation is matched with an annotated cell ground truth. Each segmented cell is matched in order of confidence, based on intersection over union (IoU) thresholds. We computed metrics based on the bounding box (rectangle defined by x and y coordinates encompassing a segmentation) and the segmentation mask predictions. AP50 and AP75 are computed using IoU thresholds 0.5 and 0.75 to match cell prediction and ground truth, respectively; and AP represents an average across multiple IoU thresholds, ranging from 0.50 to 0.95 at an interval of 0.05.

Our experimental results for nuclei instance segmentation are shown in Figure 8. Our method using the Mask R-CNN model significantly outperformed both Cellpose and SAM across all metrics for

Table 6. Key hyperparameters for Mask R-CNN model used for benchmark study

Hyperparameter	Value
Model	Mask R-CNN
Dataset	SRH550
Image transformation	Random flipping
Batch size	2 patches
Optimizer	AdamW
Learn rate	2.0E-4
Learn rate scheduler	Cosine decay
Training length	20 epochs

both bounding box prediction and instance segmentation. We attribute this to SRH being a new imaging modality whose images are out of domain for these state of the art methods.

To ensure our Mask R-CNN model performs well across all tumor types, we further benchmarked the model with different tumor types. Each method's instance segmentation AP50 performance is represented in Figure 9. We see that state-of-the-art methods perform poorly on normal tissue samples, but our method performs consistently well across tumor types. We attribute this to normal tissue samples being out of distribution for state-of-the-art methods because these images contain fewer cells.

Our dataset enabled us to train segmentation beyond nuclei, and we present preliminary results for additional cell types and structures in Figure 10. Sample of the instance segmentations from each model as well as the ground truth annotation is shown in Figure 11. While our method performed best on nuclei, we believe other classes are limited by relatively low representation within our data. Future work will focus on accounting for this limitation and increasing training data size.

LIMITATIONS

The ELUCIDATE Platform and the DetectSRH library provide an initial pipeline for annotating and training deep segmentation models on SRH data. However, the process of launching and running these tools requires both hardware resources and software expertise that may make it difficult for the lay user to engage with this protocol. Furthermore, once the software is launched, the initial annotation step is still a manual, time-consuming process that requires careful coordination with multiple users to be done efficiently. We also rely on manual transfer of the annotation data from the web platform to personal hardware for conducting model training and iterative improvement. We hope to streamline this in the future with in-browser automated segmentation and model development.

From a data standpoint, the current ELUCIDATE platform is restricted to 300×300 pixel SRH patches and does not support whole slide image (WSI) visualization or annotation due to technical constraints that would lead to tradeoffs in memory and processing requirements. WSI processing requires substantially increased memory footprints (600× increase, going from 196 KB to 120 MB per image) that reduce system scalability for collaborative environments. In addition, WSI visualization demands pyramidal, multi-resolution storage formats and specialized viewers such as OpenSeadragon that are not compatible with the current implementation of the Django-Labeller framework underlying ELUCIDATE's annotation interface. Furthermore, for single-cell segmentation tasks, patch-based annotation provides sufficient spatial context while maintaining computational efficiency, as cell boundaries and subcellular structures are adequately captured within 300×300 pixel regions at SRH resolution. While WSI support remains a desirable long-term goal, the current patch-based system provides an optimal balance between annotation precision, computational feasibility, and collaborative accessibility for the intended single-cell analysis applications.

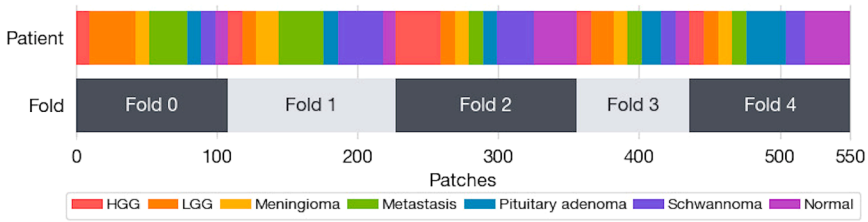


Figure 7. Distribution of patient and corresponding tumor type across the five-folds used for training and validation

The dataset showcased in the protocol is also limited to six CNS tumor types and normal brain tissue. At this time, we do not include more rare tumors. While the underlying software can be modified to view other types of histology data, it would need to be implemented by the user.

The statistical and model evaluation experiments discussed in the protocol only demonstrate the performance of a Mask-RCNN trained on ELUCIDATE's annotation data versus one-shot inference of the popular off-the-shelf models (SAM and Cellpose). More modern models that use vision transformer (ViT) architectures, such as Detection Transformer (DETR),[7] have shown state-of-the-art object detection results. However, these methods are typically trained on large quantities of data when compared to Convolutional Neural Network (CNN) architectures. For example, DETR uses 220× more training data compared to the dataset annotated in this work. Future work can incorporate these powerful models when more annotated data becomes available. In the meantime, ELUCIDATE and DetectSRH provide an accessible, data-efficient strategy that achieves state-of-the-art performance for SRH single-cell detection.

TROUBLESHOOTING

Problem 1

Issues with launching the ELUCIDATE web platform using Docker.

Potential solution

When attempting to set up ELUCIDATE for the first time, users may encounter Docker-related or Django web platform configuration issues. The build process might stall or fail with error messages such as "no such service", "failed to build", or "connection refused". After running docker-compose up, users might see containers exit immediately or the web interface remain inaccessible at localhost:8000. These issues typically stem from Docker Compose configuration problems or missing prerequisites. Common causes include incorrectly modified.env files, missing database environment variables, incorrect file paths in the compose file, or database port conflicts with existing PostgreSQL installations.

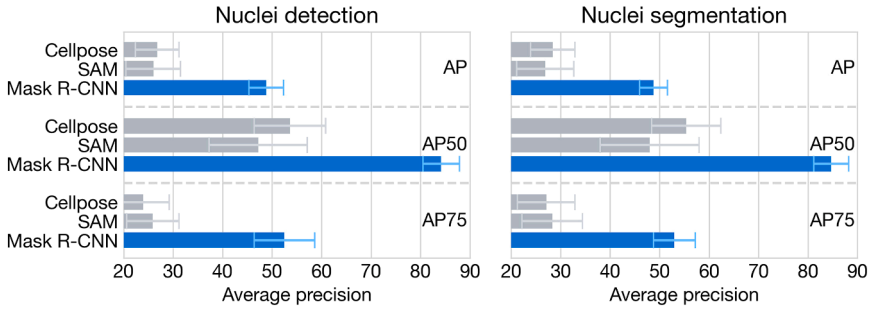


Figure 8. Comparative performance of nuclei detection and segmentation models

Performance metrics (AP, AP50, AP75) for SAM, Cellpose, and Mask R-CNN. Error bars represent ± 1 standard deviation from 5-fold cross-validation.

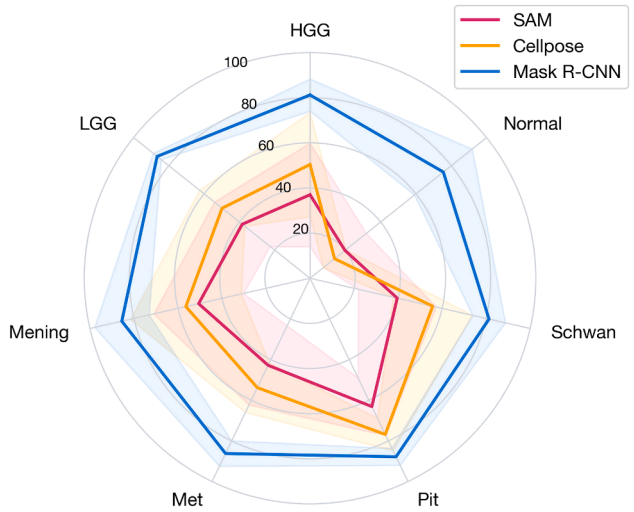


Figure 9. Nuclei instance segmentation AP50 for all methods across all tumor types

The shaded area represents ±1 standard deviation across 5 fold cross-validation. Abbreviations: HGG, high grade glioma; LGG, low grade glioma; Mening, meningioma; Mets, metastasis; Pit, pituitary adenoma; Schwan, schwannoma; Normal, normal brain.

Steps For Resolving Issue.

- Verify minimal system requirements (detailed specifications outlined in materials and equipment section):
  - Minimum 8 GB RAM available.
  - At least 64 GB free disk space.
  - Docker Desktop installed and running.
  - Docker Compose version 2.0 or higher.
- Check and fix Docker resources:

```
```bash
# Check Docker disk usage
docker system df
# Clean up unused containers and images
docker system prune -a
# Verify Docker service is running
sudo systemctl status docker
```

- Verify presence of configuration files:

```
```bash
# Check all required files exist
ls .env* local.yml
# Verify environment variables
cat .envs/local/.django cat .envs/local/.postgres
```

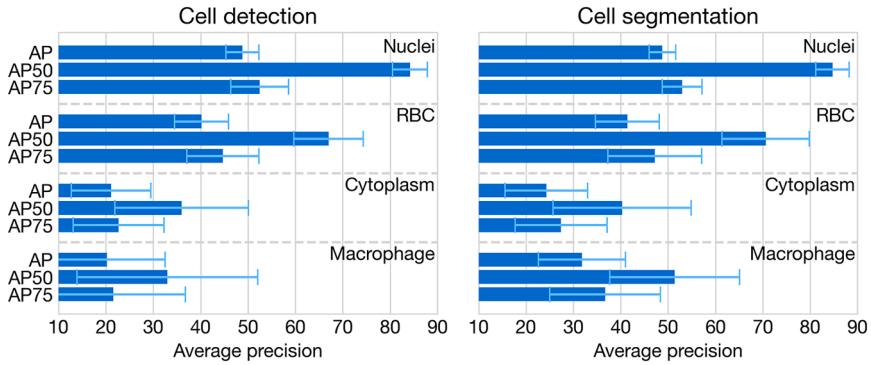


Figure 10. Mask R-CNN instance segmentation performance metrics

Detection and segmentation average precision (AP) scores across cell types. Error bars represent ±1 standard deviation from 5-fold cross-validation. AP scores shown for nuclei, red blood cells (RBC), cytoplasm, and macrophages.

Problem 2

SRH dataset not appearing in the main browser of ELUCIDATE after upload.

Potential solution

Common causes for this issue relate to the structure and format of the uploaded **.zip** file. The web portal requires all images be of TIFF format (300 × 300 pixel) containing only 2 channels and follow a strict naming convention: *[patient_id]_[slide_id]_[additional_characters].tif[f].* The patient id can be any character and slide_id must be a numerical value. These can be arbitrary as long as they follow the format and are separated by underscores.

Additionally, your zip file must follow a minimum structure: a root folder containing either all images directly or if including annotations, the JSON files must be in a separate 'annotations' subfolder in the root directory.

Steps For Resolving Issue.

- Check SRH image requirements:
  - TIFF files (suffix can be either.tif or.tiff).
  - 300 × 300 pixels.
  - 2-channel images.
  - Follows naming convention*: [patient_id]_[slide_id]_[additional_characters].tif[f].*
- Verify the folder structure by unzipping the uploaded file and check if it follows the minimum format:

```
uploaded_folder.zip /
  [patient_id]_[slide_id]_file0.tiff
  [patient_id]_[slide_id]_file1.tif
  [patient_id]_[slide_id]_file2.tif
  ..
  annotations/
    [patient_id]_[slide_id]_file0.json
    [patient_id]_[slide_id]_file1.json
```

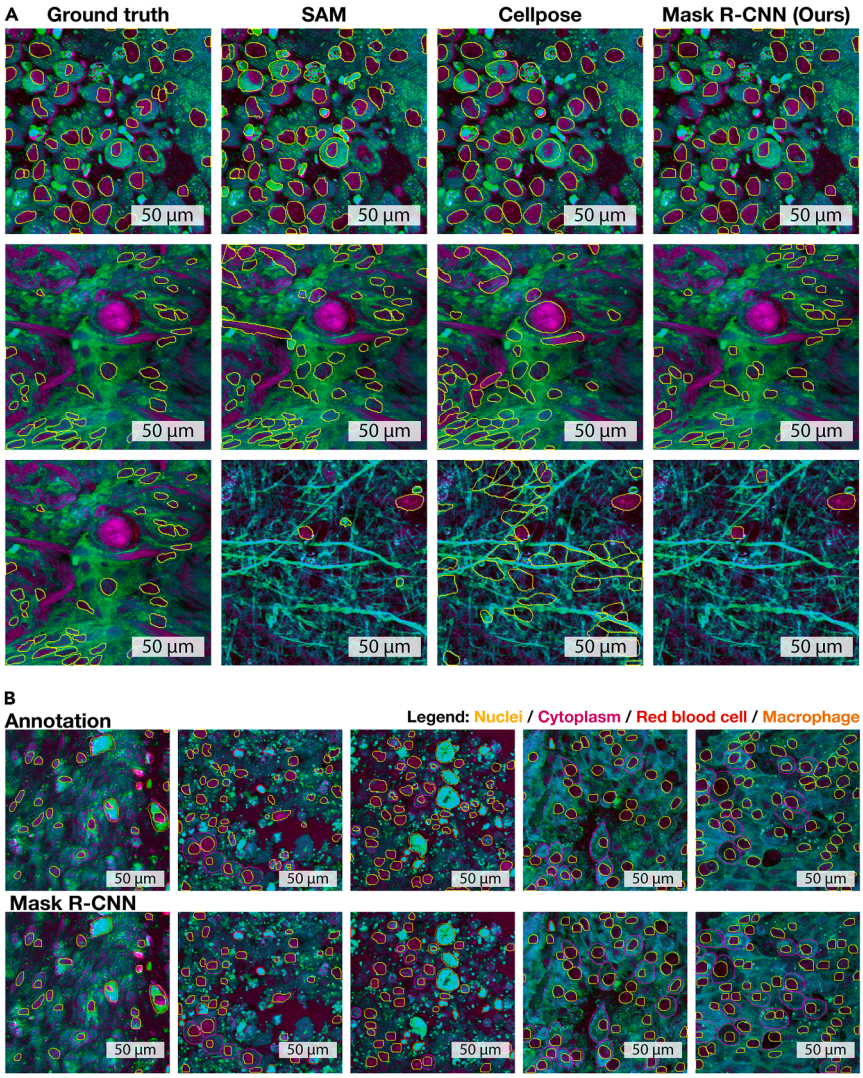


Figure 11. Model segmentation performance on SRH images

(A) Comparison of nuclei segmentation results from SAM, Cellpose, and Mask R-CNN models against ground truth annotations.

(B) Mask R-CNN segmentation examples of nuclei, cytoplasm, red blood cells, and macrophages with corresponding ground truth annotations.

```
    [patient_id]_[slide_id]_file2.json

    ...
```

- If using annotation files:
  - JSON files must match image names exactly.
  - Place in 'annotations' subfolder.
  - Follow correct JSON format (see export annotations section).
- If the dataset contains a large number of images, give several minutes for processing and refresh the browser window accordingly.

Note: A single folder containing all images is also acceptable, but organizing by patient improves dataset management.

Problem 3

SRH data with pre-existing annotations uploaded to ELUCIDATE, but segmentations not appearing in the annotation browser.

Potential solution

Most often segmentations fail to appear in the annotation browser because of label class names in JSON files that have not been defined on the ELUCIDATE platform via the administration panel. Other causes also include improperly formatted JSON annotation files, missing "default" schema in the system, as well as coordinate system mismatches in the polygon definitions.

Steps For Resolving Issue.

- Check label class setup:
  - Navigate to admin panel (/admin).
  - Verify "default" schema exists.
  - Confirm label classes match JSON files.
  - Check label class identifiers exactly match those in JSON.
- If using model predictions:
  - Verify prediction JSON matches ELUCIDATE format.
  - Check coordinate system (0-300 range).
  - Confirm all required fields present.

Common fixes:

- Re-create schema named "default".
- Add missing label classes.
- Ensure label class identifiers match JSON exactly.
- Export a working annotation for format reference.

Problem 4

Docker commands may fail depending on install version and method.

Potential solution

Docker transitioned to using "docker compose" instead of "docker-compose" with the release of Docker Compose V2, in April 26, 2022. Additionally, depending on how Docker is installed, you may need to run the provided commands with "sudo".

Steps for Resolving Issue.

- Replace "docker-compose" with "docker compose".
- Prepend your command with "sudo", so that you get "sudo <provided docker command>".

Problem 5

Unable to train the Mask R-CNN model using the DetectSRH library. The model's training loss does not decrease as expected.

Potential solution

Training can fail to progress due to issues including improper hyperparameters (such as learning rate, pixel space transformations), incorrect data formatting (such as bounding box format and image normalization), or mislabeled annotations (such as incompatible data types and class labels). If the training curve remains flat or fluctuates without improvement, adjustments to these factors are necessary.

Steps for Resolving Issue:

- Adjust the Learning Rate:
  - If too high, reduce it by a factor of 2 to prevent oscillations.
  - If too low, increase it by a factor of 2 and use a learning rate scheduler (as employed in the default configuration)
- Adjust pixel space transformations.
  - If the model is overfitting, increase the strength of transformations by employing color jittering, gaussian blur, and gaussian noise.
  - If training is unstable, consider using weak augmentations only (such as flipping).
- Validate data formatting.
  - Ensure bounding boxes follow the correct format for your model. Possible formatting include "xyxy", "xywh", and "cxcywh". The default configuration uses "xyxy" for Mask R-CNN training.
  - Verify class labels match expected values and are properly indexed. The class labels should range from 0 to number of object types. In Mask R-CNN, 0 is the default to represent the background class.
  - Check for data type mismatches in images and annotations. Pixel values may be of type uint8, but should be converted to float before training.
- Inspect the dataset.
  - Confirm all images have corresponding annotations.
  - Visualize samples to ensure bounding boxes and masks align correctly.
  - Temporarily disable augmentations to rule out transformation errors. Make sure any spatial transformations (such as flipping, random resized crop) are also applied to the bounding boxes and masks.
- Check training logs.
  - Look for nan errors, vanishing or exploding gradients.
  - Ensure there are no warnings about missing or corrupted data.

RESOURCE AVAILABILITY

Lead contact

Further information, general requests, and data access questions should be directed to and will be fulfilled by the lead contact, Abhishek Bhattacharya (abhishek.bhattacharya@nyulangone.org).

Technical contact

Further information and requests regarding the ELUCIDATE annotation platform code should be directed to the technical contact, Abhishek Bhattacharya (abhishek.bhattacharya@nyulangone.org). Further information and requests regarding the DetectSRH ML platform should be directed to the technical contacts, Eric Landgraf (elandg@umich.edu) and Cheng Jiang (chengjia@umich.edu).

Materials availability

This study did not generate new unique reagents.

Data and code availability

Source code for ELUCIDATE program can be found here: https://github.com/MLNeurosurg/elucidate/. Source code for DetectSRH library can be found here: https://github.com/MLNeurosurg/detectsrh/. SRH550 dataset and Mask-RCNN model weights trained on the entire SRH550 segmentation data can be found here: https://mlins.org/elucidate/.

ACKNOWLEDGMENTS

We would like to thank Karen Eddy for her administrative support and data collection efforts.

This work was supported, in part, by the National Institutes of Health (NIH) grants F31NS135973 (C.J.), T32GM141746 (C.J.), and K12NS080223 (T.C.H.). This work was also supported, in part, by the Chan Zuckerberg Foundation (CZI) Advancing Imaging Through Collaborative Project grant (T.C.H.), the Cook Family Brain Tumor Research Fund (T.C.H.), the Mark Trauner Brain Research Fund (T.C.H.), the Zenkel Family Foundation (T.C.H.), Ian's Friends Foundation (T.C.H.), and the UM Precision Health Investigators Awards grant program (T.C.H.).

AUTHOR CONTRIBUTIONS

A.B., E.L., C.J., and T.C.H. were responsible for conceiving the study design. A.B. created and hosted the ELUCIDATE web platform. A.C. assisted in the hosting of the ELUCIDATE platform. A.B., E.L., C.J., A.C., A.K., E.S.H., X.H., and T.C.H.

annotated the SRH500 single-cell instance segmentation benchmark. E.L., C.J., and A.B. implemented the DetectSRH library. E.L., C.J., and A.B. trained machine learning model for single-cell instance segmentation. A.B., E.L., C.J., and T.C.H. performed statistical analysis of the annotation and machine learning results. L. Wang was responsible for the acquisition of SRH images. L. Walsh assisted in SRH image acquisition. A.B., E.L., and C.J. drafted the manuscript. All authors were involved in the editing and reviewing of data and manuscript versions.

DECLARATION OF INTERESTS

T.C.H. is a shareholder of Invenio Imaging, Inc., a company developing SRH microscopy systems.

REFERENCES

1. Freudiger, C.W., Min, W., Saar, B.G., Lu, S., Holtom, G.R., He, C., Tsai, J.C., Kang, J.X., and Xie, X.S. (2008). Label-free biomedical imaging with high sensitivity by stimulated Raman scattering microscopy. Science *322*, 1857–1861. https://doi.org/10.1126/science.1165758.

2. Hollon, T.C., Pandian, B., Adapa, A.R., Urias, E., Save, A.V., Khalsa, S.S.S., Eichberg, D.G., D'Amico, R.S., Farooq, Z.U., Lewis, S., et al. (2020). Near real-time intraoperative brain tumor diagnosis using stimulated Raman histology and deep neural networks. Nat. Med. *26*, 52–58. https://doi.org/10.1038/s41591-019-0715-9.

3. Hollon, T., Jiang, C., Chowdury, A., Nasir-Moin, M., Kondepudi, A., Aabedi, A., Adapa, A., Al-Holou, W., Heth, J., Sagher, O., et al. (2023). Artificial-intelligence-based molecular classification of diffuse gliomas using rapid, label-free optical imaging. Nat. Med. *29*, 828–832. https://doi.org/10.1038/s41591-023-02252-4.

4. Jiang, C., Chowdury, A., Hou, X., Kondepudi, A., Freudiger, C.W., Conway, K., Camelo-Piragua, S., Orringer, D.A., Lee, H., and Hollon, T.C. (2022). OpenSRH: Optimizing brain tumor surgery using intraoperative Stimulated Raman histology. Adv. Neural Inf. Process. Syst. *35*, 28502–28516. https://doi.org/10.48550/arXiv.2206.08439.

5. Stringer, C., Wang, T., Michaelos, M., and Pachitariu, M. (2021). Cellpose: A Generalist Algorithm for Cellular Segmentation. Nat. Methods *18*, 100–106. https://doi.org/10.1038/s41592-020-01018-x.

6. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., et al. (2023). Segment Anything. Preprint at arXiv. https://doi.org/10.48550/arXiv.2304.02643.

7. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. In Computer Vision – ECCV 2020, A. Vedaldi, H. Bischof, T. Brox, and J.M. Frahm, eds. (Springer), pp. 213–229. https://doi.org/10.1007/978-3-030-58452-8_13.